SEKE2022

Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering

July 1 to 10, 2022
KSIR Virtual Conference Center
Pittsburgh, USA

DOI: 10.18293/SEKE2022

Proceedings preparation, editing and printing are sponsored by KSI Research Inc.

PROCEEDINGS

SEKE 2022

The 34th International Conference on Software Engineering & Knowledge Engineering

Sponsored by

KSI Research Inc. and Knowledge Systems Institute Graduate School, USA

Technical Program

July 1 – 10, 2022

KSIR Virtual Conference Center, Pittsburgh, USA

Organized by

KSI Research Inc. and Knowledge Systems Institute Graduate School, USA

ISBN: 1-891706-54-3

ISSN: 2325-9000 (print)

2325-9086 (online)

DOI reference number: 10.18293/SEKE2022

Publisher Information:

KSI Research Inc. and Knowledge Systems Institute Graduate School

156 Park Square

Pittsburgh, PA 15238 USA

Tel: +1-412-606-5022

Fax: +1-847-679-3166

Email: seke@ksiresearch.org

Web: http://ksiresearchorg.ipage.com/seke/seke22.html

Proceedings preparation, editing and printing are sponsored by KSI Research Inc. and Knowledge Systems Institute Graduate School, USA.

Printed by KSI Research Inc. and Knowledge Systems Institute Graduate School

SEKE 2024

The 34th International Conference on Software Engineering & Knowledge Engineering

July 1 – 10, 2022

KSIR Virtual Conference Center, Pittsburgh, USA

Conference Organization

CONFERENCE CHAIR

Kazuhiro Ogata, JAIST, Japan; Conference Chair
Lan Lin, Ball State University, USA; Conference Co-Chair

PROGRAM COMMITTEE CHAIR AND CO-CHAIR

Rong Peng, Wuhan University, China; PC Chair
Carlos Eduardo Pantoja, Federal Center for Technological Education, Brazil; PC Co-Chair
Pankaj Kamthan, Concordia University, Canada; PC Co-Chair

STEERING COMMITTEE CHAIR

Shi-Kuo Chang, University of Pittsburgh, USA

STEERING COMMITTEE

Vic Basili, University of Maryland, USA
Bruce Buchanan, University of Pittsburgh, USA
C. V. Ramamoorthy, University of California, Berkeley, USA

ADVISORY COMMITTEE

Raul Garcia Castro, Universidad Politecnica de Madrid, Spain
Jerry Gao, San Jose State University, USA
Swapna Gokhale, University of Connecticut, USA
Xudong He, Florida International University, USA
Natalia Juristo, Universidad Politecnica de Madrid, Spain
Taghi Khoshgoftaar, Florida Atlantic University, USA
Guenther Ruhe, University of Calgary, Canada

Masoud Sadjadi, Florida International University, USA
Du Zhang, California State University, USA

PROGRAM COMMITTEE

Silvia Acuna, Universidad Autonoma de Madrid, Spain
Shadi Alawneh, Oakland University, USA
Hyggo Almeida, Federal University of Campina Grande, Brazil
Dionysios Athanasopoulos, Queen's University of Belfast, United Kingdom
Doo-Hwan Bae, Korea Advanced Institute of Science and Technology, Korea
Kyungmin Bae, Pohang University of Science and Technology, Korea
Kuhaneswaran Banujan, Sabaragamuwa University of Sri Lanka, Sri Lanka
Vita Barletta, University of Bari, Italy
Fevzi Belli, University of Paderborn, Germany
Ateet Bhalla, Consultant, India
Swapan Bhattacharya, Jadavpur University, India
Tanmay Bhowmik, Mississippi State University, USA
Michael Bosu, Centre for Information Technology, New Zealand
Ivo Bukovsky, Czech Technical University in Prague, Czech Republic
Guoray Cai, Penn State University, USA
Juan Cano-DeBenito, Universidad PolitÃ©cnica de Madrid, Spain
Rafael Cardoso, University of Liverpool, United Kingdom
John Castro, Universidad de Atacama, Chile
Lingwei Chen, Pennsylvania State University, USA
Wen-Hui Chen, National Taipei University of Technology, Taiwan
Xiang Chen, Nantong University, China
Xiaohong Chen, East China Normal University, China
Meiru Che, University of Texas at Austin, USA
Nacha Chondamrongkul, Mae Fah Luang University, Thailand
Lawrence Chung, University of Texas at Dallas, USA
Andrea Cimmino, Universidad Politecnic de Madrid, Spain
Patrick Cook, Texas Tech Universityy, USA
Andrea DeLucia, University of Salerno, Italy
Lin Deng, Towson University, USA
Wei Dong, National University of Defense Technology, China
Bowen Du, University of Warwick, United Kingdom
Hector Duran-Limon, Universidad de Guadalajara, Mexico
Abdelrahman Elfaki, University of Tabuk, Saudi Arabia
Iaakov Exman, Jerusalem College of Engineering, Israel
Onyeka Ezenwoye, Augusta University, USA
Yuan Fei, Shanghai Normal University, China
Maria Francesca Costabile, University of Bari, Italy
Asadullah Galib, Michigan State University, USA
Jianbo Gao, Peking University, China
Honghao Gao, ShangHai University, China
RaÃºl Garcia-Castro, Universidad PolitÃ©cnica de Madrid, Spain
Ignacio Garcia Rodriguez DeGuzman, University of Castilla-La Mancha, Spain
Swapna Gokhale, Univ. of Connecticut, USA
Wolfgang Golubski, Zwickau University of Applied Sciences, Germany
Jorge Marx Gomez, University of Oldenburg, Germany
Desmond Greer, Queen's University Belfast, United Kingdom
Xudong He, Florida International University, USA
Kiyoshi Honda, Osaka Institute of Technology, Japan
Dou Hu, National Computer System Engineering Research Institute, China
Sayem Imtiaz, Iowa State University, USA
Bassey Isong, North-West University, South Africa
Clinton Jeffery, New Mexicon Tech, USA
Shuyuan Jin, Sun Yat-sen University, China

Kazi Sultana, Montclair State University, USA
Xin Sun, Ball State University, USA
Meng Sun, Peking University, China
Yanchun Sun, Peking University, China
Yutian Tang, ShanghaiTech University, China
Chuanqi Tao, Nanjing University of Science and Technology, China
Jeff Tian, Southern Methodist University, USA
Zhenzhou Tian, Xi'an University of Posts and Telecommunications, China
Fadel TourÃ©, UQTR, Canada
Christelle Urtado, LGI2P Ecole des Mines d'Ales, France
Dalton Valadares, IFPE Caruaru, Brazil
Sylvain Vauttier, Ecole des mines d'Ales, France
Gleifer VazAlves, Federal University of Technology - Parana (UTFPR), Brazil
Gennaro Vessio, University of Bari, Italy
JosÃ© Viterbo, Fluminense Federal University (UFF), Brazil
Jiaojiao Wang, China Communication University of Zhejiang, China
Zhongjie Wang, Harbin Institute of Technology, China
Jiacun Wang, Monmouth University, USA
Jian Wang, Wuhan University, China
Ye Wang, Zhejiang Gongshang University, China
Hironori Washizaki, Waseda University, Japan
Lingwei Wei, Chinese Academy of Sciences, China
Michael Weiss, Carleton University, Canada
Franz Wotawa, TU Graz, Austria
Ji Wu, Beihang Universityy, China
Peng Wu, Institute of Software, Chinese Academy of Sciences, China
Xi Wu, The University of Sydney, Australia
Lai Xu, Bournemouth University, UK
Haiping Xu, University of Massachusetts Dartmouth, USA
Frank Xu, University of Baltimore, USA
Koji Yamamoto, Fujitsu Laboratories Ltd., Japan
Guowei Yang, Texas State University, USA
Huiqun Yu, East China University of Science and Technology, China
Jiang Yue, Fujian Normal University, China
Dongjin Yu, Hangzhou Dianzi University, China
Fiorella Zampetti, University of Sannio, Italy
Pengcheng Zhang, Hohai University, China
Du Zhang, Macau University of Science and Technology, China
Yong Zhang, Tsinghua University, China
Yongxin Zhao, East China Normal University, China
Yongjie Zheng, California State University at San Marcos, USA
Nianjun Zhou, IBM, USA
Huibiao Zhu, East China Normal University, China
Hongming Zhu, Tongji University, China
Eugenio Zimeo, University of Sannio, Italy

BEST DEMO AWARD COMMITTEE

Kuhaneswaran Banujan, Sabaragamuwa University of Sri Lanka; Chair, Best Demo Award, Sri Lanka

PUBLICITY CO-CHAIRS

Patrick Cook, Texas Tech University, USA; Publicity Co-Chair
Michael Bosu, New Zealand; Publicity Co-Chair

ASIA LIAISON

Hironori Washizaki, Waseda University, Japan

EUROPE LIAISON

Raul Garcia Castro, Universidad Politecnica de Madrid, Spain

INDIA LIAISON

Swapan Bhattacharya, National Institute of Technology Karnataka, Surathakl, India

Foreword

Welcome to the 34th International Conference on Software Engineering and Knowledge Engineering (SEKE), in KSIR Virtual Conference Center, Pittsburgh, PA, USA. In the last 30 years, SEKE has established itself as a major international forum to foster, among academia, industry, and government agencies, discussion and exchange of ideas, research results and experience in software engineering and knowledge engineering. The SEKE community has grown to become a very important and influential source of ideas and innovations on the interplays between software engineering and knowledge engineering, and its impact on the knowledge economy has been felt worldwide. On behalf of the Program Committee, it is our great pleasure to invite you to participate in the technical program of SEKE.

This year, we received 170 submissions. Through a rigorous review process where a majority of the submitted papers received three reviews, and the rest with two reviews, we were able to select 67 full papers for the general conference (39 percent),50 short papers (30 percent), 5 demos (3 percent) and 48 rejects (28 percent). SEKE 2022 Technical Program consists of one keynote session, 11 paper presentation sessions, 2 panel sessions and one demo session. We greatly appreciate the committee members and authors of accepted papers in professional roles to serve as the chairs of the technical sessions.

The high quality of the SEKE 2022 technical program would not have been possible without the tireless effort and hard work of many individuals. First of all, we would like to express our sincere appreciation to all the authors whose technical contributions have made the final technical program possible. We are very grateful to all the Program Committee members whose expertise and dedication made our responsibility that much easier. Our gratitude also goes to the keynote speaker who graciously agreed to share his insight on important research issues, to the conference organizing committee members for their superb work, and to the external reviewers for their contribution.

Last but certainly not the least, we must acknowledge the important contributions that the KSI staff members have made. Their timely and dependable support and assistance throughout the entire process have been truly remarkable. Finally, we wish you have productive discussion, great networking and effective virtual presentation to participate in SEKE 2022.

Rong Peng, Wuhan University, China; PC Chair
Carlos Eduardo Pantoja, Federal Center for Technological Education, Brazil; PC Co-Chair
Pankaj Kamthan, Concordia University, Canada; PC Co-Chair

Keynote

The Essential Structure of Software

Daniel Jackson, PhD
Massachusetts Institute of Technology
USA

Abstract

Structure is at the heart of everything we do in software. And yet we don't have a compelling answer to the most obvious question: what is the structure of an app? We have user interface structure (e.g. pages and components), and we have internal structure (modules and subsystems). But we don't have a way to describe the essential structure, which would reveal what an app does, how it compares to other apps, and how a user should make sense of it.

In this talk, I'll present a radical new way to think about software structure in terms of concepts: independent and reusable units of dynamic functionality that can be combined in flexible ways. I'll show how this perspective helps ground familiar intuitions but also exposes new insights.

About the Speaker

Daniel Jackson is professor of computer science at MIT, and associate director of CSAIL. For his research in software, he won the ACM SIGSOFT Impact Award, the ACM SIGSOFT Outstanding Research Award and was made an ACM Fellow. He is the lead designer of the Alloy modeling language, and author of the book Software Abstractions. He chaired a National Academies study on software dependability, and has collaborated on software projects with NASA on air-traffic control, with Massachusetts General Hospital on proton therapy, and with Toyota on autonomous cars. His new book, The Essence of Software, was recently published.

Panel Session ESEKE: Education and SEKE

Co-Organizers and Co-Chairs: Pankaj Kamthan and Hironori Washizaki

Theme: Educational and Professional Implications of SWEBOK

Description: Software continues to play an indispensable role in society, not least due to the increased reliance on digitization during the COVID-19 pandemic. Software Engineering has evolved in several different directions since it was established as a discipline over 50 years ago. Much has changed in the ensuing decades, especially how Software Engineering is viewed and pursued, and SWEBOK Version 4 aims to reflect this evolution.

In particular, this panel session will address pressing issues around the following selection of topics:

(1) The Need for Evolving SWEBOK from Version 3 to Version 4
(2) The Prospects and Challenges of Integrating SWEBOK in Software Engineering Curriculum
(3) Usefulness of SWEBOK for Software Practitioners in the Industry

The session comprises an esteemed group of panelists from academia and industry, specializing in multiple different areas of Software Engineering, thereby bringing much desired multidisciplinary perspective and diverse experience to the aforementioned topics. The panelists are:

(1) **David Parsons** <david@themindlab.ac.nz> (The Mind Lab, New Zealand), Specialties: Agile Education, Agile Practices, Software Engineering Education
(2) **Nazlie Shahmir** <nazlie_shahmir@cpr.ca> (Canadian Pacific Railway Limited, Calgary, Canada), Specialties: DevOps, Software Project Management, Software Quality, Software Requirements, Software Testing
(3) **Steve Schwarm** <schwarm@ieee.org> (Retired/Part-Time Synopsys - Black Duck Software, USA), Specialties: Specialties: Software Construction, Standards, Quality, and Programming Languages
(4) **Yatheendranath TJ** <y.tarikere@ieee.org> (DhiiHii Laboratories Private Limited, Bengaluru, India), Specialties: Computing Foundations, Software Engineering Management and Training
(5) **Steve Tockey** <steve.tockey@construx.com> (Construx Software, USA), Specialties: Software Engineering Economics, Software Quality, Software Requirements
(6) **Hironori Washizaki** <washizaki@waseda.jp> (Waseda University, Tokyo, Japan), Specialties: Machine Learning Software Engineering, Software Design, Software Engineering Education, Software Maintenance and Evolution, Software Quality Assurance, Software Reuse

Moderator: Pankaj Kamthan <kamthan@cse.concordia.ca> (Concordia University, Montreal, Canada), Specialties: Agile Methodologies, Conceptual Modeling, Software Engineering Education, Software Patterns, Software Quality, Software Requirements

Each topic will be allotted 20 minutes. At the end of the panel discussion, the floor will be opened to the audience, welcoming any questions that they might have.

Panel Session CSQS: Conceptual Software and Quantum Software

Chair and Moderator: Prof. Iaakov Exman, Jerusalem College of Engineering, Israel

Panelists: Prof. Daniel Jackson, MIT, USA
Prof. Jonathan Aldrich, CMU, USA
Prof. Harold Thimbleby, Swansea University, UK

Description: A high-quality panel should be thought provoking: to trigger questions, without any sort of definitive answers. The role of participants and of the Panel audience is to ask even more questions providing at most half-baked answers, leaving second thoughts for home. Thus, this Panel is restricted to a very short time duration. It all started with a research proposal from MIT's Daniel Jackson, dated 2013 [1], from which we learned Fred Brooks' idea [2] that "*Conceptual Integrity* is the most important consideration for software system design". It culminated with the recent thought provoking and excellent book by Daniel Jackson on "The Essence of Software – Why Concepts matter for Great Design" [3]. Yes, books should also be thought provoking. We wrote *culminated* very hesitantly, since it gives the false impression that this is the end of the story. As an antidote here are some trigger questions

Questions (Q) and Half-Baked Thoughts (HBT): As an antidote here are some trigger questions and half-baked thoughts.

Q1 – Concepts matter for great design *or* Concepts are the very essence of Software?
HBT1- Focusing on apps' design implies that concepts are not the very essence of Software; Software is something else besides concepts. Our remark: "*yes* and *no*". More about this, later on.
Q2 – Is programming essential to Software? What is *the* programming language of Software?
HBT2- Emphatically *NO*. The natural language of software is the natural language of humans. Therefore, the Software vocabulary is constantly changing, together with novel technologies.
Q3 – Is Software concepts' integrity enough to avoid the dire consequences of aviation or healthcare accidents? Particularly in cases where software has been proven to be faulty?
HBT3- Probably *NOT*. Any additional ideas?
Q4 – Software Modularity should assure Conceptual Integrity? Algebraic representation of software, and its newborn child "Quantum Software Models", manipulate vectors to be modular, without touching concepts at all. How is it possible?
HBT4- The Panel audience may decide to concisely discuss it, or may prefer to read papers [4],[5].

References:

[1] Daniel Jackson, "Conceptual Design of Software: A Research Agenda", MIT-CSAIL-TR-2013-020 August 2013.
[2] Fred Brooks, *The Design of Design*, Addison-Wesley, Boston, MA, 2010.
[3] Daniel Jackson, *The Essence of Software*, Princeton University Press, Princeton, NJ, 2021.
[4] Iaakov Exman and Alon T. Shmilovich, "Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design", 2021 IEEE/ACM 2nd Int. Workshop on Quantum Software Engineering (Q-SE), within ICSE'2021, http://arxiv.org/abs/2103.13755, 2021.
[5] Iaakov Exman and Alexey Nechaev, "Quantum Software Models: Software Density Matrix is a Perfect Direct Sum of Module Matrices", in Proc. SEKE'2022, pages 434-439, this Conference, 2022.

Table of Contents

Notes: (S) denotes a short paper. (D) denotes demo description.

**** This paper was withdrawn by the authors due to errors in the paper.*

A Systematic Mapping Study of Information Retrieval Approaches Applied to Requirements Trace Recovery

Bangchao Wang[1,2], Heng Wang[2], Ruiqi Luo[1,2*], Sen Zhang[2], Qiang Zhu[1,2],
[1] Engineering Research Center of Hubei Province for Clothing Information, Wuhan Textile University, Wuhan, China
[2] School of Computer Science and Artificial Intelligence, Wuhan Textile University, Wuhan, China

***Abstract*— Context: Requirements trace recovery (RTR) is always time-consuming, tedious, and fallible. There has been a growing interest in applying information retrieval (IR) to automate the process of recover trace links between requirements artifacts and other software artifacts. Objective: In this review, our objective is to identify the state-of-the-art of how IR has been explored to automate RTR and provide an overview of the research at the intersection of these two fields. Method: A systematic mapping study has been conducted, searching the main scientific databases. The search retrieved 1587 citations and 34 articles are retained as primary studies. Results: The results show the most active authors and publication distribution. It presents four kinds of IR models and 21 enhancement strategies applied to perform RTR. Besides, the lists of 37 experimental datasets and 9 measures, commonly used together to evaluate IR-based RTR approaches, are provided. Conclusions: Vector Space Model (VSM) and Latent Semantic Index (LSI) are the most two studied IR models used in RTR. CoEST becomes the most popular, convenient and stable source of datasets. Precision and Recall are the most common measures used to evaluate the performance of IR methods. Overall, IR-based RTR is becoming an increasingly mature cross research field.**

Keywords*— *requirements trace recovery, information retrieval, systematic mapping study

I. INTRODUCTION

Requirements Traceability (RT) is defined as 'the ability to describe and follow the life of a requirement in both a forward and backward direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases)' [1]. Requirements trace recovery (RTR), as an important activity in RT, can help software engineers discover dependencies that exist between requirements artifacts and other software artifacts, evaluate the requirements coverage rate and calculate the influence of requirements change etc. [2].

In recent years, most extensive efforts have been devoted to studying information retrieval (IR) based RTR approaches in RT research community [2]. By introducing IR, it greatly alleviates the problems of heavy manual workload, difficult maintenance, and error-prone problems faced by traditional approaches [2].

This systematic mapping study (SMS) provides evidence-based insights that can help researchers gain a good understanding of IR-based RTR approaches. We believe that readers interested in RT can use this paper as a map for finding studies relevant to their situation.

The remainder of this paper is organized as follows. Section II analyzes the related works, Section III reports the details of our research methodology and logistics, Section IV provides the main findings from our SMS, and Section V discusses the validity threats. Finally, Section VI concludes this study.

II. RELATED WORKS

We found only two systematic reviews related to IR-based RTR [3][4]. The one by Saleem M et al. [3] focus on surveying whether term mismatch is a real barrier for IR-based RT approaches. Besides, this review summarizes the approaches that attempt to solve the term mismatch problem. Since these summarized approaches are only a part of the whole IR-based traceability approaches. Therefore, this review cannot provide an overview of the research at the intersection of IR and RTR.

The other one by Borg M et al. [4] surveys the state-of-the-art of IR-based software traceability (ST). Since ST has a wider research scope than RT, some IR-based RTR approaches are presented in this review. However, the time interval of the primary studies in [4] is before 2012, which indicates that more recent progress in IR-based RTR approaches has not yet been summarised.

Overall, these works do not provide the recent overview of the research at IR-based RTR. Therefore, to the best of our knowledge, there is no SMS on the status of the IR-based RTR. Our work covers this gap.

In this review, our objective is to identify the state-of-the-art of how IR has been explored to automate RTR and provide an overview of the research at the intersection of these two fields.

III. RESEARCH METHOD

We have conducted an SMS (which is a well-defined and rigorous method) to identify and interpret relevant studies regarding a particular research question, topic area, or phenomenon of interest [5]. The goal of an SMS is to provide a fair, credible, and unbiased evaluation of a research topic using a trustworthy, rigorous, and auditable method. Hence, SMS is an appropriate method for our research, which is aimed at identifying the overall status of IR-based RTR approaches.

A. Research questions

Many relevant studies on IR-based RTR approaches appeared during 2012–2021, and recent progress has not been summarised. To identify the overall status of IR-based RTR, we defined the following research questions (RQs):

RQ1. What are the authors/venues of the primary studies?

RQ2. Which IR models and enhancement strategies have been applied to perform RTR?

RQ3. Which datasets have been applied to verify IR-based RTR approaches?

RQ4. Which measures have been applied to evaluate the performance of IR-based RTR approaches?

B. Search process

We design an SMS protocol to guide the search process based on the SMS guidelines [5]. Relevant papers are

* Corresponding author: Ruiqi Luo (Email: rqluo@wtu.edu.cn).
DOI reference number: 10.18293/SEKE2022-098

retrieved automatically from the databases.

1) Inclusion and exclusion criteria.

Once the potentially relevant studies have been obtained, their actual relevance needs to be assessed. We defined the following inclusion and exclusion criteria to select studies from the search results based on the SMS guidelines [5].

Inclusion criteria:

I1: The time span of the study is 2012.1—2021.12.

I2: The research topic of study must be IR-based RTR.

I3: The study is not a review paper.

I4: The papers are written in English.

I5: When two papers with the same technology and topic are provided by the same author, we select the one that is described in greater detail.

Exclusion criteria:

E1: The time span of the study is not during 2012.1—2021.12.

E2: The research topic of study is not IR-based RTR.

E3: The study is a review paper.

E4: The paper is not written in English.

E5: When two papers with the same technology and topic are provided by the same author, we exclude the one that is described less thoroughly.

2) Search scope.

Time period. We specify the time period of the published studies for this SMS from January 2012 to December 2021, which is when we started this SMS.

Electronic databases. Based on the suggestion in [5] and the access authority of our institution, the following databases are selected as the primary study sources: IEEE, Google Scholar, Elsevier, EI Compendex, and Springer.

3) Search terms.

We use population, intervention, comparison, and outcome (PICO) criteria to define the search terms for database search based on the SMS guidelines [5]. Table I shows the search terms in population and intervention.

TABLE I. LIST OF SEARCH TERMS IN POPULATION, INTERVENTION

PI	Search terms
Population(P)	requirement traceability, requirement trace, requirement tracing, requirement traceability recovery
Intervention (I)	information retrieval, IR, semantic

Population: The population in this SMS is 'Requirements Traceability'. We use words that are synonymous to RT as the population (e.g., 'Requirements Tracing', 'Requirements Trace', and 'Requirements Trace Recovery').

Intervention: The intervention is 'information retrieval'. We use the word 'information retrieval', 'IR' and 'semantic' as the intervention.

Comparison: Because there is no comparative approach for this review according to the SMS guidelines [5], the part of the comparison specified in PICO is not considered in the construction of search terms.

Outcome: Because there is no outcome for this review according to the SMS guidelines [5], the part of the outcome specified in PICO is not considered in the construction of search terms.

4) Study search and selection.

Fig. 1 summarizes the study search and selection results. The overall process consists of the following phases.

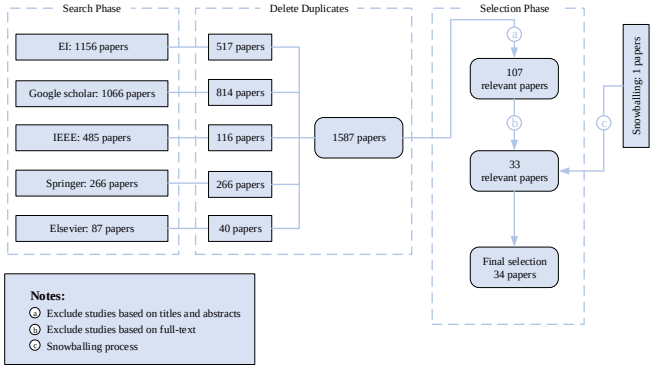


Fig. 1. Study search and selection results.

Phase 1: Keyword-based literature retrieval

The search terms (defined in Table I) are applied only to the title, keyword, and abstract because a full text search would yield a large number of irrelevant results.

Phase 2: 1st round of literature filtering

The titles, abstracts, and keywords of all potential primary studies are checked by the second and fourth authors against inclusion and exclusion criteria. If it is difficult to determine whether a paper should be included or not, it is reserved for the next phase.

Phase 3: 2nd round of literature filtering

In this round, the first and third authors read the full text to determine whether the paper should be included or not based on the inclusion and exclusion criteria. When an agreement cannot be reached, they are asked to state the reasons for inclusion/exclusion to an arbitration panel.

Phase 4: Snowballing

After the filtering, 'snowballing' is conducted to find omitted papers. We adopt the snowballing process proposed by Claes Wohlin [6] to iteratively search the reference list and papers cited in a study until no new papers are found. Then, the new papers are checked against the inclusion and exclusion criteria.

C. Data extraction and synthesis

When conducting data extraction, the authors carefully read the primary studies and have conducted a strict peer-review process. Before official extraction, a pilot of the data extraction has been performed. During official extraction, data are extracted based on a detailed set of questions. We kept a record of the extracted information in a spreadsheet for later analysis.

For synthesizing data, qualitative and quantitative data is involved. When synthesizing the data, this process was supported by the extracted data spreadsheet mentioned above.

After performing a separate analysis of the qualitative and quantitative results, we investigate the combination of both sources of evidence. Additionally, we also explore the combination of results from different research questions to build an evidence map that identifies research trends and gaps according to multiple perspectives (questions-answers).

IV. RESULTS

This section presents the results of the SMS and is arranged in the order of the research questions presented in Section III-A.

A. RQ1. What are the authors/venues of the primary studies?

Nasir Ali	5	Nan Niu	4	Anas Mahmoud	4
Jiale Zhou	3	Bangchao Wang	2	Rong Peng	2
Danissa V. Rodriguez	2	Doris L. Carver	2	Hongyu Kuang	2
Yue Lu	2	Jyoti	2	Others(>20)	1

Fig. 2. Authors of the primary studies.

Fig. 2 shows the author distribution. More than 100 authors have appeared in the 34 primary studies. Nasir Ali, Nan Niu, Anas Mahmoud are the top 3 most active authors. It should be noted that some authors that appear only once are not presented in Fig. 2 because of the space limit.

REC 4 ICPC 2 Others(28) 1

Fig. 3. Conferences and Journals in which the studies are published.

Thirty-four primary studies have been published in 30 types of conferences and journal, as shown in Fig. 3. The RE conference (REC), and ICPC are the top 2 places where the most papers are published.

B. RQ2. Which IR models and enhancement strategies have been applied to perform RTR?

The general process of IR-based trace recovery is shown in Fig. 4, which usually includes three stages: **1) Preprocessing Stage:** both the source (Requirements Artifact) and target artifacts (Software Artifact) are regarded as text, and noise information is removed through certain document preprocessing methods to generate document representation that is convenient for subsequent processing; **2) Trace Links Generation (Recovery) stage:** Calculating the similarity between the two artifacts using various document similarity calculation methods, sorting according to the similarity scores, and selecting the candidate trace links according to the set threshold. **3) Trace Link Refinement Stage:** The candidate trace links are refined by manual or semi-automatic methods, and the trace links are finally confirmed by the analyst.

Usually, after preprocessing, trace links can be automatically established using various types of IR-based models. As shown in Table II, Vector Space Model (VSM) and Latent Semantic Index (LSI) are the most two commonly used IR-based models in trace link generation. It should be noted that some studies make the IR model as an integral part of the whole approaches, such as [26][31][33]. Another notable conclusion is that more than half primary studies have presented, used or verified more than one IR models.

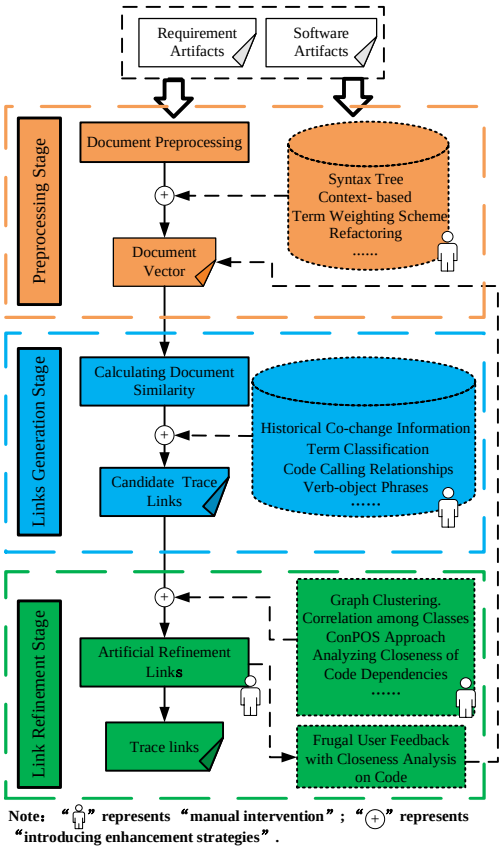


Fig. 4. The general process of IR-based trace recovery

To improve the performance, various types of enhancement strategies are proposed. For example, as shown in Fig. 4, Syntax Tree [27] and Refactoring [15][32][36] are used to reduce the adverse effects caused by inconsistent terminology or missing, misplaced signs in textual artifacts. The details of enhancement strategies for IR-based RTR are listed in Table III. From this table, most strategies focus on how to improve the performance of the VSM and LSI.

TABLE II. IR MODELS USED IN REQUIREMENTS TRACE RECOVERY

IR Models	Algebraic Models	VSM	[7] [8] [10] [11] [13] [14] [15] [18] [19] [20] [21] [22] [23] [24] [25] [27] [28] [29] [32] [34] [35] [36] [37] [38] [39] [40]
		LSI	[9] [11] [12] [14] [15] [16] [17] [23] [30] [32] [35] [36] [38] [40]
	Probabilistic Models	JS	[11] [22] [24] [39] [40]
		TM	[21] [29]
	Other		[26] [31] [33]

Note: VSM: Vector Space Model, LSI: Latent Semantic Index, JS: Jensen-Shannon Model, TM: Topic Model.

C. RQ3. Which datasets have been applied to verify IR-based RTR approaches?

The answer for this RQ can be used as a guideline for the researchers to select datasets based on their research needs and the characteristics of the datasets. For instance, for each of these datasets, Table IV provides a link to the open-source datasets, along with other meta-data details associated with it, such as primary studies that used it, trace space (the maximum counts of trace links), and other characteristics of the dataset.

TABLE III. LIST OF ENHANCEMENT STRATEGIES FOR IR-BASED REQUIREMENTS TRACE RECOVERY APPROACHES

Strategy	IR model				Applying Phrase	Strategy Characteristics
	VSM	LSI	JS	TM		
Context- based [8][10][37]	●				**P**	Separating intent from context in requirements
Improved Term Weighting Scheme [12][16]		●			**P**	Proposing an improved term weighting scheme, namely, Developers Preferred Term Frequency/Inverse Document Frequency (DPTF/IDF)
Refactoring [15][32][36]	●	●			**P**	Solving the problem of missing symbols, misplaced symbols and repeated symbols
Syntax Tree [27]	●				**P**	Primary identifier keywords are converted to comment keywords by their similarity in appearance in the syntax tree location
Verb-object Phrases [7]	●				**P**	Extracting verb-object phrases as main information and essential meaning
Analyzing Close Relations [13]	●				**G**	Calculating the close relations (semantic similarity) between target artifacts
Term Classification [17][30]		●			**G**	Categorizing class names, comments, and all other terms in code
Model-Driven Engineering (MDE) [19]	●				**G**	Combining use of both MDE and IR, analyzing the textual information (organization and hierarchy) contained in the model to retrieve implicit links between documents
Hybrid Method [21][29]	●			●	**G**	Combing VSM and BTM which can help relieve data sparsity caused by short text
Genetic Algorithm [29]				●	**G**	Configuring initial parameters of BTM by introducing Genetic Algorithm
Code Calling Relationships [20]	●				**G**	Identifying errors between requirements and code traces by code-calling relationships
Historical Co-change Information [23]	●	●			**G**	Taking the processed corpora and co-change information of classes as input to reorder and filter baseline links
Dynamic Integration of Structural and Co-change Coupling [28]	●				**G**	Retrieving indirect links based on weighted integration of structural coupling and class coupling based on change history
Configuration Management Log [35][38]	●	●			**G**	Restoring links by finding revisions in the configuration management log that contain words related to requirements
Frugal User Feedback with Closeness Analysis on Code [40]	●	●	●		**R**	Introducing only a small amount of user feedback into the closeness analysis on call and data dependencies in code
User Feedback [35]	●				**R**	Introducing user validation for candidate links to improve accuracy
Analyzing Closeness of Code Dependencies [11]	●	●	●		**R**	Quantifying the interaction degree of call dependency and data dependency between two code classes
Class Clustering [17]		●			**R**	The products in the clustering have similar trace relationships
Correlation among Classes [25]	●				**R**	Using structural or co-changing dependencies or both to find correlations between classes and use these dependencies to verify traceability links
Graph Clustering [34]	●				**R**	Information about the cohesion of artifacts within a level of refinement helps improve the trace retrieval process between levels of refinement
ConPOS approach [39]	●		●		**R**	Pruning trace links using the primary POS classification and apply constraints to recovery as a filtering process

Note: " ● " represents support; "P" represents Preprocessing Stage, "G" represents Links Generation Stage, "R" represents Links Refinement Stage.

TABLE IV. DATASETS' INFORMATION AND THE STUDIED PAPERS WHICH USED THE DATASETS

Dataset Name	Source Artifacts (Number)	Target Artifacts (Number)	Space	True Links	Scale	Freq.	Resource links	Reference
iTrust	Use cases (34)	Code (243)	8262	603	Large	16	http://www.coest.org/	[8][11][12][13][14][15][16][23][25][26][28][30][32][36][39][40]
	Requirements (50)	Code (299)	14950	314				
eTour	Use Cases (58)	Code (116)	6728	308	Large	11	http://www.coest.org/	[7][8][14][15][18][24][29][30][31][32][36]
	Requirements (58)	Code (116)	6728	366				
EasyClinic	Requirements (30)	Code (47)	1410	93	Small	6	http://www.coest.org/	[7][9][13][21][29][34]
	Use Cases (30)	Test Cases (63)	1890	63				
CM-1	High-level Requirements (235)	Low-level Requirements (220)	51700	4050	Large	5	http://www.coest.org/	[9][13][14][30][34]
	High-level Requirements (235)	Design (220)	51700	361				
	Requirements (235)	Design (220)	51700	361				
	Requirements (235)	Use Case (Unclear)	Unclear	Unclear	Unclear			
Pooka	Requirement (298)	Code (90)	26820	546	Large	5	http://www.suberic.net/pooka/	[12][16][22][23][39]
EBT	Requirements (41)	Test Cases (25)	1025	51	Small	4	http://www.coest.org/	[18][21][24][29]
	requirements (40)	Code (50)	2000	98				
GanttProject	Requirements (16)	Code (124)	1173	315	Small	4	http://www.ganttproject.biz	[11][13][20][40]
Albergate	requirements (17)	Code (55)	935	54	Small	3	http://www.coest.org/	[18][24][31]
SIP Communicator	Requirements (82)	Code (1771)	145222	871	Large	3	http://www.jitsi.org	[22][23][39]
WARC	Non-functional Requirements (21)	Software Requirements Specification (89)	1869	58	Small	2	http://www.coest.org/	[21][29]
	Functional Requirements (43)	Software Requirements Specification (89)	3827	78	Large			
GANNT	High-level Requirements (17)	Low-level Requirements (69)	1173	68	Small	1	http://www.coest.org/	[13]
jEdit v4.3	Requirements (34)	Code (483)	16422	Unclear	Large	1	http://www.jedit.org.	[22]
Infinispan	Requirements (237)	Code (388)	91956	1515	Large	1	http://infinispan.org/	[40]
Lucene	Requirements (116)	Code (413)	47908	744	Large	1	http://lucene.apache.org	[12]
Rhino v1.6	Requirements (268)	Code (138)	36984	Unclear	Large	1	http://www.mozilla.org/rhino/	[22]
Lynx	Requirements (90)	Code (298)	26820	507	Large	1	http://lynx.isc.org/	[39]
Maven	Requirements (36)	Code (94)	3384	155	Large	1	http://maven.apache.org/	[40]
Pig	Requirements (68)	Code (236)	16048	356	Large	1	https://pig.apache.org/	[40]
Mylyn	Requirements (Unclear)	Code (Unclear)	Unclear	Unclear	Unclear	1	http://www.eclipse.org/mylyn/developers	[26]

Note: There are 18 datasets that cannot be obtained, i.e., Chess [20], CUnit [38], iBooks [7], iRobot [10], iTruck [10], iSudoku [10], jHotDraw (JHD) [11][20], SMS [7], MODIS [9], MR0 [9], MR1 [9], MR2 [9], Pine [13], VideoOnDemand (VoD) [20], WDS [30][36], Waterloo [34], LEDA [31], network control system [38].

Thirty-seven datasets used for experimental validation were extracted from thirty-four primary studies, as shown in Table IV. The iTrust, eTour and EasyClinic are the top 3 most popular datasets, which all are provided by the Center of Excellence for Software & Systems Traceability (CoEST). Besides, In the nineteen open-source datasets, eight datasets are provided by CoEST, and the resource can be found at http://www.coest.org/. These nineteen open-source datasets have been used sixty-eight times. More than 70% (48/68, 70.6%) of cases, the used datasets are from CoEST. Obviously, it becomes the most popular, convenient and stable source of datasets.

On the other side, twelve primary studies (12/34, 35.3%) use the eighteen non-open-source datasets, as shown at the bottom of Table IV. It prevents researchers from reproducing experiments and using datasets.

TABLE V. NUMBER OF USED DATASETS DISTRIBUTION FOR PRIMARY STUDIES

Number of datasets used in primary studies	Number of primary studies	Proportion
0	2	5.88%
1	6	17.65%
2	5	14.71%
3	12	35.29%
4	5	14.71%
5	3	8.82%
>5	1	2.94%
Total	34	100%

As shown in Table V, the fact that nearly a quarter (8/34, 23.5%) of primary studies validated their methods by applying to less than two datasets is a threat to the external validity [6]. The more datasets used, the more general the method becomes. To mitigate external threats, researchers are encouraged to use various types of datasets as many as possible.

RQ4. Which measures have been applied to evaluate the performance of IR-based RTR approaches?

After trace links are generated, the evaluation of them is an indispensable task. In most cases, precision and recall are the most common measures used to evaluate the performance of IR-based RTR approaches, as shown in Table VI. Once further evaluation is needed, the secondary measures, such as MAP, AP, DiffAR, Lag, Selectivity and Cliff's Delta, are good candidates. Due to the space limit, no specific details to the performance measures, which are used to evaluated IR-based RTR approaches, are given here.

TABLE VI. PERFORMANCE MEASURES USED IN PRIMARY STUDIES

Categories	Measures	Primary Studies
Primary Measure	Recall	[7][10][11][12][13][16][17][18][20][21] [22][23][24][25][26][27][28][29][30][31] [32][34] [35][36][37][38][39][40]
	Precision	
	F-Measure	[7][12][22][25][27][28][29][35][38]
Secondary Measure	MAP	[8][9][10][11][13][14][32][33][34][36] [39][40]
	AP	[11][13][33][34][39][40]
	DiffAR	[14][32][36]
	Lag	[14]
	Selectivity	[29]
	Cliff's Delta	[40]

V. VALIDITY THREATS DISCUSSION

In this section, we aim to discuss these potential threats that influence the data extraction and the findings of this SMS. According to the guidelines for analyzing the validity threats to SE methods and processes [6], conclusion validity, construct validity, internal validity, external validity will be discussed in the following.

Conclusion validity: It is possible that some papers excluded by this review should have been included. To mitigate this type of threat, the selection process and the inclusion and exclusion criteria described in Section III-B are carefully designed and discussed by authors to minimize the risk of exclusion of relevant studies.

Construct validity: The main constructs in this SMS are 'requirements traceability' and 'information retrieval'. We respectively use these two terms and their synonyms to ensure that all selected primary studies are relevant to the intersection of these two fields. Meanwhile, snowballing from literature sources is performed complementary to database search to ensure that relevant studies are covered as much as possible.

Internal validity: In this SMS, a different participant may end up with different data extraction and analysis results. This may be a threat to the internal validity. To mitigate the threat, the data extraction is performed collaboratively by two authors. Moreover, any conflicts are discussed and resolved by all the authors in this process.

External validity: It is concerned with establishing the generalizability of the SMS results, which is related to the degree to which the primary studies are representative of the review topics. To mitigate external threats, the search process described in Section III-B is defined after several trial searches. Moreover, we have tested the coverage and representativeness of retrieved papers.

VI. CONCLUSIONS

RTR is always heavy manual workload, time-consuming, tedious, and fallible [2]. To alleviate these problems, there has been a growing interest in applying IR to automate the process of recover trace links between requirements artifacts and other software artifacts [2]. In this SMS, we survey the state-of-the-art of how IR has been explored to automate RTR and provide an overview of the research at the intersection of these two fields. By analyzing the 34 primary studies, the following conclusions have been obtained:

Firstly, VSM, LSI, JS and TM are the main kinds of IR models applied to perform RTR during the decade. Besides, 21 enhancement strategies are developed to improve the performance of these models. Researchers are encouraged to use multiple strategies to construct a combination approach.

Secondly, CoEST is proved to be the most popular, convenient and stable source of datasets. Researchers are encouraged to use various types of open-source datasets as many as possible. It helps other researchers to reproduce experiments and validate the proposed approaches.

Thirdly, Precision and Recall are the most common measures used to evaluate the performance of IR methods. Also, MAP is the most popular secondary measures. Researchers are encouraged to evaluate IR methods from different dimensions by applying different measures.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 62102291.

REFERENCES

[1] O. Gotel and C. W. Finkelstein, "An analysis of the requirements traceability problem," Proceedings of IEEE International Conference on Requirements Engineering,1994, pp. 94-101.

[2] B. Wang, R. Peng, Y. Li, et al.,"Requirements traceability technologies and technology transfer decision support: A systematic review," Journal of Systems and Software, 2018, 146: 59-79.

[3] M. Saleem,and N. M. Minhas ,"Information retrieval based requirement traceability recovery approaches-a systematic literature review," University of Sindh Journal of Information and Communication Technology, 2018, 2(4): 180-188.

[4] M. Borg, P. Runeson and A. Ardoe, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability,"Empirical Software Engineering, 2014.

[5] P. Kai, S. Vakkalanka and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update, " Information and Software Technology, 2015, 64:1-18.

[6] C. Wohlin, P. Runeson, M. Host, et al.,"Experimentation in Software Engineering," Springer-Verlag, Berlin, Heidelberg, 2012.

[7] Y. Zhang, C. Wan and B. Jin, "An empirical study on recovering requirement-to-code links," 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD),pp.121-126.

[8] J. Zhou, Y. Lu and K. Lundqvist, "An Improved VSM-based Post-Requirements Traceability Recovery Approach Using Context Analysis," 2013

[9] S. Eder, H. Femmer, B. Hauptmann and M. Junker, "Configuring Latent Semantic Indexing for Requirements Tracing," 2015 IEEE/ACM 2nd International Workshop on Requirements Engineering and Testing, 2015, pp. 27-33.

[10] J. Zhou, Y. Lu and K. Lundqvist, "A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems," 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, 2013, pp. 252-259.

[11] H. Kuang, J. Nie, H. Hu, et al., "Analyzing closeness of code dependencies for improving IR-based Traceability Recovery". 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 68-78.

[12] N. Ali, Z. Sharafi, Y. G. Gueheneuc, et al., "An empirical study on the importance of source code entities for requirements traceability," Empirical Software Engineering, 2015 ,20(2):442-478.

[13] H. Wang, G. Shen, Z. Huang, et al.,"Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery," Frontiers of Information Technology & Electronic Engineering, 2021,22(7):957-968.

[14] A. Mahmoud, N. Niu and S. Xu, "A semantic relatedness approach for traceability link recovery," 2012 20th IEEE International Conference on Program Comprehension (ICPC), 2012, pp. 183-192.

[15] F. Faiz, R. Easmin and A. U. Gias, "Achieving Better Requirements to Code Traceability: Which Refactoring Should Be Done First?," 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC), 2016, pp. 9-14.

[16] N. Ali, Z. Sharafl, Y. -G. Guéhéneuc and G. Antoniol, "An empirical study on requirements traceability using eye-tracking," 2012 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 191-200.

[17] J. Shao, W. Wu and P. Geng, "An Improved Approach to the Recovery of Traceability Links between Requirement Documents and Source Codes Based on Latent Semantic Indexing," 13th International Conference on Computational Science & Its Applications, Springer Berlin Heidelberg, 2013.

[18] D. V. Rodriguez and D. L. Carver, "An IR-Based Artificial Bee Colony Approach for Traceability Link Recovery," 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), 2020, pp. 1145-1153.

[19] N. Sannier and B. Baudry, "Toward multilevel textual requirements traceability using model-driven engineering and information retrieval," 2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE), 2012, pp. 29-38.

[20] A. Ghabi and A. Egyed, "Code patterns for automatically validating requirements-to-code traces," 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, 2012, pp. 200-209.

[21] B. Wang, R. Peng, Z. Wang, et al., "An Automated Hybrid Approach for Generating Requirements Trace Links," 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019.

[22] N. Ali, Guéhéneuc, Yann-Gal, and G. Antoniol, "Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links," IEEE Transactions on Software Engineering, 2013, 39(5):725-741.

[23] N. Ali, F. Jaafar and A. E. Hassan, "Leveraging historical co-change information for requirements traceability," 2013 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 361-370.

[24] D. V. Rodriguez and D. L. Carver, "Multi-Objective Information Retrieval-Based NSGA-II Optimization for Requirements Traceability Recovery," 2020 IEEE International Conference on Electro Information Technology (EIT), 2020, pp. 271-280.

[25] Jyoti and J. K. Chhabra, "Filtering of false positives from IR-based traceability links among software artifacts," 2017 2nd International Conference for Convergence in Technology (I2CT), 2017, pp. 1111-1115.

[26] P Hu¨Bner. "Quality Improvements for Trace Links between Source Code and Requirements," REFSQ-2016 Workshops, co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2016.

[27] S. Nagano, Y. Ichikawa and T. Kobayashi, "Recovering Traceability Links between Code and Documentation for Enterprise Project Artifacts," 2012 IEEE 36th Annual Computer Software and Applications Conference, 2012, pp. 11-18.

[28] Jyoti and J. K. Chhabra, "Requirements Traceability Through Information Retrieval Using Dynamic Integration of Structural and Co-change Coupling," International Conference on Advanced Informatics for Computing Research. Springer, Singapore, 2017.

[29] B. Wang, R. Peng, Z. Wang, et al., “An Automated Hybrid Approach for Generating Requirements Trace Links,” International Journal of Software Engineering and Knowledge Engineering, 2020.

[30] N. Niu and A. Mahmoud, "Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited," 2012 20th IEEE International Requirements Engineering Conference (RE), 2012, pp. 81-90.

[31] A. Ghannem, M. S. Hamdi, M. Kessentini and H. H. Ammar, "Search-based requirements traceability recovery: A multi-objective approach," 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1183-1190.

[32] A. Mahmoud and N. Niu, "Supporting requirements traceability through refactoring," 2013 21st IEEE International Requirements Engineering Conference (RE), 2013, pp. 32-41.

[33] R. Jain, S. Ghaisas and A. Sureka, “SANAYOJAN: a framework for traceability link recovery between use-cases in software requirement specification and regulatory documents,” 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE 2014.

[34] P. Rempel, P. Mäder and T. Kuschke, "Towards feature-aware retrieval of refinement traces," 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), 2013, pp. 100-104.

[35] R. Tsuchiya, H. Washizaki, Y. Fukazawa, et al., "Interactive Recovery of Requirements Traceability Links Using User Feedback and Configuration Management Logs," 27th International Conference on Advanced Information Systems Engineering, CAISE 2015, Springer International Publishing, 2015.

[36] A. Mahmoud and N. Niu, "Supporting requirements traceability through refactoring," 2013 21st IEEE International Requirements Engineering Conference (RE), 2013, pp.32-41.

[37] J. Zhou, "Requirements development and management of embedded real-time systems," 2014 IEEE 22nd International Requirements Engineering Conference(RE), 2014, PP. 479-484.

[38] R. Tsuchiya, H. Washizaki, Y. Fukazawa, et al., "Recovering Traceability Links between Requirements and Source Code Using the Configuration Management Log," IEICETransactions on Information and Systems, 2015, 98-D(4).

[39] N. Ali, H. Cai , A. Hamou-Lhadj , et al., "Exploiting Parts-of-Speech for effective automated requirements traceability," Information and software technology, 2019, pp.126-141.

[40] H. Kuang, H. Gao, H. Hu, et al., "Using Frugal User Feedback with Closeness Analysis on Code to Improve IR-Based Traceability Recovery," 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), 2019, pp. 369-379.

A framework for Requirements specification of machine-learning systems

Xi Wang*† and Weikai Miao‡✉
*School of Computer Engineering and Science, Shanghai University, China
†Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai 201114, China
‡Software Engineering Institute, East China Normal University, China

***Abstract*—The rapid development of machine learning (ML) systems has raised many concerns over their quality. Due to the inherent complexity and uncertainty, most of the traditional quality assurance techniques have been challenged, including requirements specification. Current strategies mainly focus on model extraction from existing neural networks to improve interpretability and facilitate system analysis, but failing to include user expectations on the system. To handle the problem, this paper proposes a specification framework for ML requirements where each ML system is regarded as a set of snapshot systems along the evolvement process. There are 3 layers in the framework and the hierarchy indicates that higher-level models need to be built based on lower-level ones. The bottom layer consists of meta snapshot model and meta data model serving as the meta models for snapshot systems and data requirements respectively. The middle layer is for snapshot models each describing a snapshot system through relations between its outputs produced with different inputs. The top layer is a learning model capturing the evolvement process by transitions among snapshot models. These transitions are activated by data models instantiated from meta data model. We adopt the specification of a self-driving system to illustrate the framework.**

I. Introduction

Machine-learning (ML) system has gained much attention recent years not only for its thrilling achievement on training intelligent machines but also for the challenging difficulties on quality assurance [1]. These difficulties mainly come from the complexity and uncertainty of ML systems since these systems would adjust their functions themselves through mechanisms difficult to be understood by human. Challenges are proposed to all kinds of traditional software quality assurance methods including requirements modeling [2].

As the key activity in the early stage of software development, requirements modeling intends to clarify and describe expected behaviors in a requirements specification which serves as guidance for implementation and basis for verification. Traditional methods are suitable for deterministic functionalities but can hardly tackle ML systems that constantly evolve and produce unpredictable results [3]. Some solutions have been proposed and most of them focus on extraction algorithms for constructing system model from ML models [4]. Since their major goal is to facilitate human understanding and semantic analysis of existing ML systems by interpreting their learning behaviors with easy-to-digest notations, user expectation on the ML systems is not involved in the derived model and the system requirements remains unclear.

Several attempts were made to deal with user requirements for ML systems. Their most concerned issue is the performance of the trained ML models and how it can be defined and measured [5]. Once these non-functional requirements are specified, adversarial examples can be accordingly designed and used as training data to re-train the ML models and improve their performance. Data set has also gained much attention as it serves as the key element for distinguishing ML systems from traditional ones. Researchers are pursuing standards for evaluating data set quality and guidelines for deriving high-qualified data set. Besides, there are a few works on supporting decision making during the requirement stage, such as the selection of ML models and activation functions. Necessary requirement information is captured through domain analysis and suggested decisions will be accordingly provided. All these attempts made significant progress in requirements modeling for ML systems from different aspects, but there is still lack of systematic method for specifying system behaviors and their relation with data sets.

To this end, this paper proposes a systematic framework for requirements specification of ML systems. It adopts feature-oriented analysis method to specify the included models based on features and explains the evolvement process of the ML system as a set of snapshot systems each representing a state of the ML system during the process. Requirements on data sets are also included as triggers for snapshot system behaviors and the transitions among them.

Specifically, the framework is composed of 3 layers and lower-level models are the basis for higher-level models. The bottom layer includes 2 meta models. One is meta snapshot model with environment and system elements for describing the environment and internal states of the ML system respectively. The other is meta data model with built-in and learning-relative elements for describing data attributes independent from and dependent on the ML system respectively. These elements are organized in feature models that illustrate their hierarchical relations.

The middle layer consists of the snapshot systems necessary to be specified. Due to the difficulty in obtaining expected outputs from a snapshot system, we turn to the output relation strategy inspired by metamorphic testing [6]. Instead of modeling system behaviors through relations between input and output, this strategy pays attention to the change of outputs caused by the change of inputs. It describes a snapshot system

DOI reference number: 10.18293/SEKE2022-0143

as a set of relations based on meta models where each relation connects two outputs produced with a input data model and its variant respectively.

The top layer is a learning model describing the learning behavior by transitions among the snapshot systems. Each transition is labeled with a data model instantiated from its meta model, indicating that the original system will be evolved into the destination system by learning from the data model. Each snapshot system can either be modeled using output relation strategy or via its relation with the original system it evolves from.

Our framework covers the important aspects of ML requirements and deals with their complicated relations with hierarchy. An example self-driving system is introduced to illustrate the technical details within each layer.

The remainder of this article is organized as follows. Section II summarizes the related work. Section III explicitly describes the requirements modeling framework for ML systems with an example self-driving system. Section IV concludes the paper and discusses some future works.

II. Related Work

There are mainly 2 kinds of modeling strategies for ML systems. The first kind extracts simpler models from ML models to enable human understanding and the application of existing analysis methods, since ML models are usually complex and difficult to be understood and analyzed. In [7], guidance from RNN's step-wise predictive decisions and context information is provided for extracting weighted automata from neural network. In [8], a method for learning an FSA from a trained RNN model is given where the RNN is first trained and an FSA is learned based on the clustering result of all hidden states. In [4], probabilistic automata is extracted from RNN in a request-specific way and the experiment shows the accuracy and scalability of the method. In [9], the knowledge hidden in pre-trained CNN is interpreted by explanatory graph where different part patterns are disentangled from each filter of CNN in an unsupervised manner. Since the above works are conducted on already implemented ML systems for revealing their behaviors from black-box, they paid little attention to user requirements on the system and lack effective mechanisms for requirements specification. By contrast, our approach focuses on requirements modeling for ML systems and provides description framework from various aspects.

The other kind aims at modeling ML requirements from certain perspectives. In [10], goal-oriented requirements analysis method is extended as evidence-driven method where many aspects of decision making remain as hypotheses until being validated or invalidated by experiments, field tests and operation. It is conducted at goal level without touching concrete functions of ML systems. In [11], a logical approach is given for specifying statistical properties of ML systems based on a Kripke model. It includes formal notations for robustness and fairness of classifiers, as well as relations among properties of classifiers. In [12], definition of fairness is provided for an effective and scalable automatic testing method of ML systems in the domain of text classification. In [13], conventional quality characteristics is extended for ML systems with its measuring method and a method for requirements identification is proposed to derive quality characteristics and measurement method. In [14], a data-driven engineering process is proposed to link the operational design domain with the requirements on data sets at different levels. These researches mainly concentrate on non-functional and data requirements of ML systems, but are still incapable of specifying their overall expected behaviors. Our framework intends to bridge the gap by specifying ML systems at 3 different layers in a systematic way to cover both snapshot and learning behaviors.

III. Framework for modeling ML systems

Comparing to traditional software which determines its behavior with programs, ML systems learn from training data set and evolve themselves towards the target goals. This fact makes it impossible to adopt traditional modeling method in specifying ML systems as it is only suitable for predictable behaviors. Customized modeling method can only be established when differences between two kinds of software systems are fully identified. If we take snapshots for a ML system along its evolving process, a set of simpler systems can be captured each representing one state of the ML system reached by learning. The transitions among these states are triggered by training data, indicating the learning path of the ML system. According to the above analysis, there are 3 aspects to be included in the requirements of ML systems: snapshot systems for solving the target problem at different time points, learning behaviors for transiting among different snapshot systems and training data for activating the learning behaviors. To enable the specification of all these 3 aspects for ML systems, a requirements modeling framework with 3 layers is given as shown in Fig. 1.

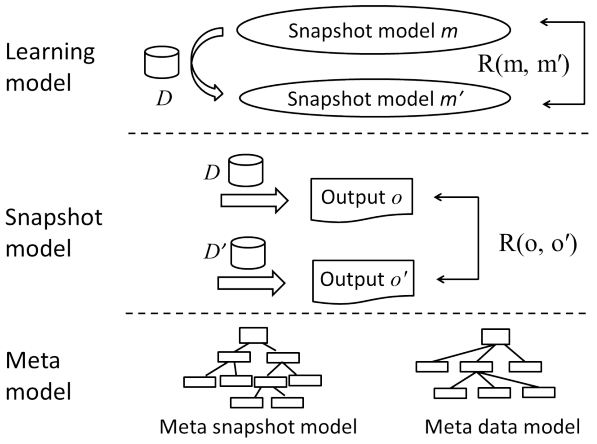


Fig. 1. The requirements modeling framework for ML systems

The hierarchy indicates that upper-level models need to be built based on lower-level ones. The bottom level consists of two meta models: meta snapshot model as meta model for

describing snapshot systems and meta data model as meta model for describing training or testing data set. They provide general patterns to build various concrete snapshot and data models for the higher layers.

The middle level adopts the idea of metamorphic relation to describe concrete snapshot systems since the expected output is almost impossible to determine. In stead of modeling the relation between input and output data as in tradition method, we turn to the relation $R(o, o')$ between expected output o and o' produced by the snapshot system with input data D and D' respectively. For each pair of different input data D and D', either corresponding concrete data models or their relations need to be established based on meta data model. With the common elements from meta snapshot model, relation $R(o, o')$ can be described as properties of relevant elements.

The top level deals with learning behaviors by describing the evolvement of the ML system based on training data sets. Although learning process is continuous, expected requirements can be reflected by important transitions each representing a measurable change of the ML system made by learning from certain kind of data set. A transition connects two snapshot models m and m' before and after the corresponding change and is labeled with a data model D instantiated from its meta model. It indicates the learning process where the ML system will be transited from snapshot system m to m' if being trained with data set D. The change made by D can also be described as a relation $R(m, m')$ between m and m' if necessary.

We will present the details of each layer with an example of self-driving system which fully controls the movement of vehicles and evolves by learning from animation on labeled route and driving scenarios.

Meta model

The bottom-level meta snapshot model and meta data model aim at capturing the basic elements for composing concrete snapshot model and data model. We adopt Feature-oriented Analysis Method to specify these elements with feature model illustrated as tree structure [15]. Each node of the tree represents an element and its children nodes decompose the element into lower-level elements. There are 4 kinds of children nodes: *mandatory* element, *optional* element, *alternative* elements where only one element can be included and *or* elements where at least one element must be included. There are 2 kinds of dependency relations among elements: *requires* indicating the inclusion of certain element requires for the inclusion of another one and *excludes* indicating that only one of the two elements can be included.

Meta snapshot model consists of environment and system elements. The former indicates the factors that would affect system behaviors from outside of the system and the latter indicates system variables determining the state of the system. Fig.2 gives the partial feature model of the environment elements for the example self-driving system. It shows 4 of the mandatory environment elements with some of their children elements: the weather condition, the local policy such as driving on left side and other governmental policies, the brightness, the road scenes such as the traffic signs erected above roads to give instructions, jam or intersection scenarios and various obstacles.

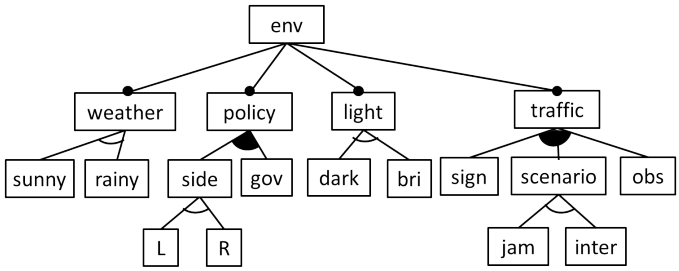


Fig. 2. The feature model for environment elements of the self-driving system

Fig.3 shows the partial feature model of the system elements for the self-driving system. To specify the system behavior, its steering angle, velocity and route are 3 of the elements that must be clarified. Two of the optional elements performance and windshield-wiper will be specified as various metrics and wiping intervals respectively. Two example metrics are given: latency of the behavior and the accuracy of object detection and trajectory prediction. The element *detect* is defined as a mapping $obj \rightarrow a$ denoting the accuracy a of detecting object obj.

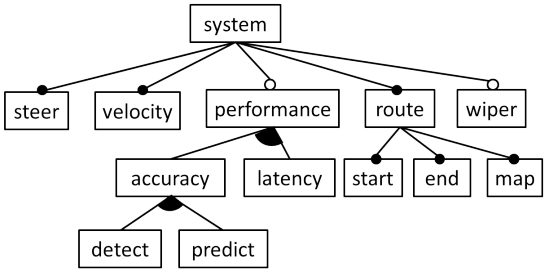


Fig. 3. The feature model for system elements of the self-driving system

Dependency relations can also be established among the above elements. For example, elements *rainy* requires for *wiper* since the windshield-wiper needs to be turned on for rainy days.

Meta data model includes built-in and learning-relative elements which represent data attributes independent from or depending on the specific learning system respectively. A concrete data model can be obtained by specifying the values of or properties on these elements. For built-in elements, one unified feature model is sufficient since they are shared by concrete data models fed to different ML systems. But for learning-relative elements, different feature models need to be built for different learning systems since their definitions are given based on the specific ML systems.

The partial feature model of built-in elements is given in Fig.4 with one of the optional elements and three of the mandatory elements. For each data set, we can decide whether to specify its collection and expire date but must at least clarify the representation of the involved data, the source of

the data and its labeling method. Four example representations are given and each representation is attached with its own attributes, such as the size and resolution of image data. With the rapid development of Multi Modal Machine Learning, more and more data sets are provided in multiple representations. Our feature model allows for multiple representations through *or* children elements of *representation* where *relation* denotes the correspondence between data in different representations. For example, multi-modal data set *nuScenes* includes images from camera, pointclouds from Lidar, the returns from Radar sensors and human annotated semantic map for the same obstacles to train self-driving systems with complementary data [16].

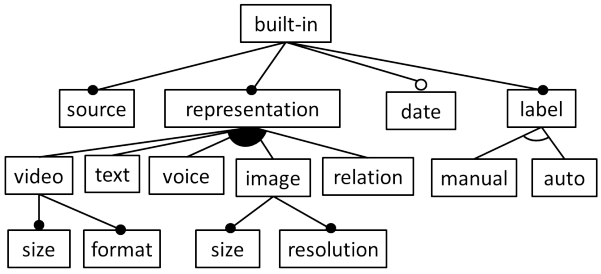


Fig. 4. The feature model of meta data model built-in elements

Learning-relative elements are created for each specific ML system since learning-relative requirements on data sets depends on what to be learned from them. Involving essential environment and system elements of functions to be learned for solving the target problem, meta snapshot model serves as the basis for achieving learning-relative elements. Fig. 5 provides a feature model template for learning-relative elements of specific ML systems. The semantic of a data set must be specified by a set of pairs $es-pair_1, ..., es-pair_n$ each $es-pair_i = (Prop_{env}, Prop_{system})$ representing an environment-system pair where $Prop_{env}$ indicates an instantiated environment model and $Prop_{system}$ indicates a concrete system model. With well-defined environment and system state, each pair corresponds to a kind of scenarios and the universal set of pairs is able to capture all the behaviors to be learned from the data set by the ML system. Optional element *metrics* represents measurements on the data set including one or more of its attributes.

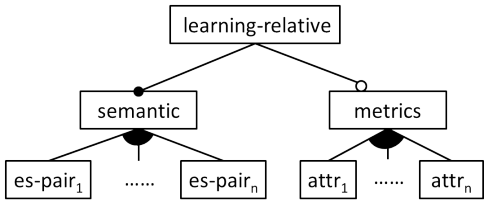


Fig. 5. A template for the feature model of meta data model learning-relative elements

For the example self-driving system, the semantic of its data sets is interpreted through property pairs on its own environment and system elements. Domain-specific attributes are attached to element *metrics*: *diversity* measuring dissimilarity among closest and furthest samples, *distribution* measuring the distribution of data samples in the context of certain features, *complexity* measuring the complexity of objects within the traffic scenes.

It should be noted that the presented example meta models are only partially given and new domain-dependent elements may need to be introduced as the relevant domain develops. These meta models should be maintained by analyst and domain experts to provide advanced information for concrete model construction.

Snapshot model

A snapshot system can be regarded as one of the states of the ML system. It takes a data set as input and transfers itself to a new system state. Since the input data provides specific settings on environment and system initialization, its data model contains only one environment-system pair.

Theoretically, the behavior of a snapshot system is deterministic because of the pause of the evolvement at that time point. However, instead of executing under man-made instructions, the snapshot system leads its own complicated implementation process with the given training data, making itself a black-box difficult to be understood and analyzed. Even at requirements stage, expected functions remain unclear because they are hidden in the training data. In most cases, we found ourselves trapped in a dilemma where solutions achieved by ML systems cannot be interpreted but need to be modeled and verified. Although we could simply define the relation between inputs and outputs in some cases, such as correctly recognizing or never crashing on a pedestrian, but such requirement contains little effective information and can hardly contribute to system verification.

Inspired by metamorphic testing, we revealed users' expectation on system behaviors when making certain change to input data and thus define snapshot model as follows.

Definition 1. Given a snapshot system S with its input domain DM_S consisting of data models, the snapshot model of S is a set $R_1^{T_1}, ..., R_n^{T_n}$ and each $R_i^{T_i}$ represents a relation formulation $\forall_{D \in DM_S} \cdot S(D^{T_i}) = [S(D)]^{T_i'}$ where T_i denotes a kind of transformational relation between data models, D^T denotes the data model obtained by performing transformation T to data model D, $S(D)$ denotes the system state of S after receiving data model D, and $[s]^T$ denotes the state achieved by conducting transformation T on the original state s.

According to the definition, each snapshot system is modeled as a set of behavior relations between different system states caused by transformations on the original input data models. For each behavior relation, the change of any input data model by the given transformation should result in system state reached from its original state with the same transformation. Transformation on data model is described based on the built-in and learning-relative elements from meta data model while transformation on system state is described based on system elements from meta snapshot model.

We will take the self-driving system as an example to illustrate the definition. Assume the system has evolved to certain snapshot version S, the following are some example relations established for the corresponding snapshot model. Note that we use $\tau(m, < e >)$ to denote the value of element e within model m and $\sigma(m, < e >)$ to denote a model different from m in the way they specify element e.

Exchange of start and end point: As one of the key functions in self-driving system, navigation component guides the vehicle from start to the end point with the generated route. It is difficult to specify our requirement on the resultant route, but we expect the fact that exchanging the start and end point should result in a new route similar to the original one under the same traffic condition. The corresponding relation formulation is given as follows.

$$\forall_{D,D^T \in DM_S} \cdot \tau(D, < start >) = \tau(D^T, < end >) \\ \wedge \tau(D, < start >) = \tau(D^T, < end >) \\ \Rightarrow dist(\tau(S(D), < map >), \tau(S(D^T), < map >)) < \epsilon$$

The difference between route map and map' is measured by function $dist(map, map')$ and the threshold ϵ needs to be given by domain expert.

Change of weather : This relation can be established for driving scenarios under different weather conditions. Although some system elements maybe specified in different ways under different weather conditions such as *wiper*, but there should not exist long distance between steering angles since steering angle largely depends on road scenes rather than weather conditions. The relation formulation is given as follows.

$$\forall_{D \in DM_S} \cdot | \ \tau(S(D), < steer >) - \\ \tau(S(\sigma(D, < weather >), < steer >) \ | < \epsilon$$

The threshold ϵ measures the distance between steering angles in the systems states led by different data models and needs to be given by domain expert.

Perturbations to traffic signs: Correct recognition of traffic signs is crucial to the safety of self-driving vehicles and perturbations to them should not affect the recognition result. This is also an important and verifiable property for the detection of pedestrians and other obstacles, we will take traffic signs as an example for illustration. The corresponding formulation is given as follows.

$$\forall_{D,D^T \in DM_S} \cdot dist(\tau(D, < sign >), \tau(D^T, < sign >)) < \epsilon \\ \Rightarrow S(D) = S(D^T)$$

If the distance between the original sign and the perturbed one is less than the given threshold ϵ, the behavior of the snapshot system should be consistent.

Learning model

Based on the definition of snapshot models, the top-level learning model can be regarded as a state transition diagram where each state represents a snapshot system and each transition represents the evolvement of the ML system activated by training data. The formal definition is given as follows.

Definition 2. The learning model of a ML system is a 4-tuple (M, s_0, DM, δ) where M is a non-empty set of snapshot models, $s_0 \in M$ is the model of the initialized ML system, DM is the universal set of involved data models and $\delta : M \times DM \rightarrow M$ is the transition function relating two snapshot models by a data model.

Connecting the identified snapshot models by transitions labeled with data models, the learning model traces the expected learning process of the corresponding ML system. Each snapshot model can either be defined as a set of relations between outputs produced by different inputs or a relation to its original model. Each data model is specified as a set of properties on its meta data model.

Fig. 6 gives a partial learning model for the self-driving system. It includes 4 snapshot models as critical learning points and 3 data models as learning materials contributing to the evolvement. The self-driving system is initially a snapshot system m_0 with all parameters of the neural network set as random values. After we train m_0 with data sets as described in D_1 or D_2, the ML system will be evolved into snapshot system m_1 or m_2. These two different transitions come from user expectation on customizing the self-driving system for specific customers and markets. Snapshot system m_1 and m_2 intend to serve the countries or areas driving on left and right respectively and their corresponding training data D_1 and D_2 should provide videos and images under different policies.

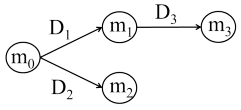


Fig. 6. An example learning model for the self-driving system

Specifically, data models D_1 and D_2 are described as a set of properties based on the pre-defined meta data model. Table I lists some example properties where $parent.child$ denotes the value of the children element *child* of element *parent*.

TABLE I
THE PROPERTIES FOR DATA MODEL D_1 AND D_2

D_1	$\forall_{es-pair_i \in \tau(D_1, <semantic>)} \cdot$ $es - pair_i.env.policy.side = L$
	$\tau(D_1, < representation >) = \{video\} \wedge$ $\tau(D_1, < video.size >) > 10000h$
	$\tau(D_1, < distribution >) =$ $P(\tau(D_1, < scenario >) = jam) > 0.7$
D_2	$\forall_{es-pair_i \in \tau(D_2, <semantic>)} \cdot$ $es - pair_i.env.policy.side = R$
	$\tau(D_2, < representation >) = \{image\} \wedge$ $\tau(D_2, < image.resolution >) > 600ppi$
	$\tau(D_2, < distribution >) =$ $P(\tau(D_2, < light >) = dark) > 0.9$

Training data in $D1$ satisfies at least 3 properties: all traffic scenarios are under the policy of driving on the left, video is the only format and the size should be more than 10000 hours, more than 70% of environment settings involve traffic jam.

Large portion of the videos are required to capture scenarios of traffic jam and the resultant system m_1 is supposed to be applied for metropolitan areas. There are also 3 example properties for D_2: all traffic scenarios are under the policy of driving on the right, image is the only format and the resolution should be more than 600 *ppi*, more than 90% of environment settings are in dark. Most of the training images are in dark environment and system m_2 is expected to support truck drivers.

The more properties we specify, the more effective training data we can collect for achieving expected snapshot systems. As the basis for property description, element structures in meta models need to be enriched through the cooperation of software and domain experts.

Snapshot model m_1 and m_2 can be built by establishing relations between output from different input, as we have mentioned in the middle layer. They are not explicitly described due to the sake of space.

Learning from data model D_3, m_1 will be further evolved into m_3. Training data in D_3 should be collected with signs satisfying the following property.

$$\forall_{es-pair_i \in \tau(D_3, <semantic>)} \cdot \forall_{sign \in es-pair_i.env.traffic.sign} \cdot \exists_{obj \to a \in \tau(m_1, <detect>)} \cdot a > 0.9 \wedge dis(sign, obj) < \varepsilon$$

According to the property, all the driving scenarios in D_3 involve adversarial samples for sign recognition. After conducting the transition from m_1 to m_3, the robustness of the system will be improved in terms of the following accuracy relation between m_1 to m_3.

$$\forall_{obj \to a \in \tau(m_3, <detect>)} \cdot a > [\tau(m_1, < detect >)](obj)$$

It formally states that the accuracy in object detection will be increased after the evolvement from snapshot system m_1 to m_3.

At requirement stage, learning model serves as a study plan for guiding the implementation of learning strategy. Meanwhile, it's detailed description on transitions among snapshot systems facilitates system traceability and maintenance.

IV. Conclusions

This paper intends to take our first step towards systematic requirements modeling method for ML systems. A framework with 3 layers is proposed where lower-level models serve as basis for high-level models. The bottom-level provides meta models for describing environment, system state and data. The middle level consists of snapshot models for describing the behaviors of the ML system at certain learning point and the top level is a learning model that describes learning behavior by transitions among snapshot models. This framework enables us to understand and analyze ML systems at requirements stage and bridges the gap between system modeling and validation.

Our case study demonstrates the framework with a partial model for simplicity. Its performance on large-scale systems needs to be evaluated by applying it in real settings. As the number of elements in meta models increases, the establishment of relevant properties will be much more difficult. How to facilitate model construction for each layer is one of our future works. Furthermore, a supporting tool also needs to be developed in the future to alleviate the burden of model construction, analysis and management.

Acknowledgment

This work is supported by the NSFCs of China (No. 61872144, No. 61902234 and No. 61872146) and National Social Science Foundation (No. 17AZX003).

References

[1] M. Chechik, "Uncertain requirements, assurance and machine learning," *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2019-Septe, pp. 2–3, 2019.

[2] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?" *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1857–1871, 2021.

[3] K. Ahmad, M. Bano, M. Abdelrazek, C. Arora, and J. Grundy, "What's up with Requirements Engineering for Artificial Intelligence Systems?" *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 1–12, 2021.

[4] G. Dong, J. Wang, J. Sun, Y. Zhang, X. Wang, T. Dai, J. S. Dong, and X. Wang, "Towards Interpreting Recurrent Neural Networks through Probabilistic Abstraction," *Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*, pp. 499–510, 2020.

[5] K. M. Habibullah and J. Horkoff, "Non-functional Requirements for Machine Learning: Understanding Current Use and Challenges in Industry," *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 13–23, 2021.

[6] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2020.

[7] X. Zhang, X. Du, X. Xie, L. Ma, Y. Liu, and M. Sun, "Decision-Guided Weighted Automata Extraction from Recurrent Neural Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11 699–11 707, 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17391

[8] B. J. Hou and Z. H. Zhou, "Learning with Interpretable Structure from Gated RNN," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2267–2279, 2020.

[9] Q. Zhang, R. Cao, F. Shi, Y. N. Wu, and S. C. Zhu, "Interpreting CNN knowledge via an explanatory graph," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 4454–4463, 2018.

[10] F. Ishikawa and Y. Matsuno, "Evidence-driven Requirements Engineering for Uncertainty of Machine Learning-based Systems," *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2020-Augus, pp. 346–351, 2020.

[11] Y. Kawamoto, *Towards Logical Specification of Statistical Machine Learning*. Springer International Publishing, 2019, vol. 11724 LNCS. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-30446-1_16

[12] P. Zhang, J. Wang, J. Sun, X. Wang, G. Dong, X. Wang, T. Dai, and J. S. Dong, "Automatic Fairness Testing of Neural Classifiers through Adversarial Sampling," *IEEE Transactions on Software Engineering*, vol. 5589, no. c, pp. 1–20, 2021.

[13] K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, L. Joeckel, J. Siebert, and J. Heidrich, "Requirements-driven method to determine quality characteristics and measurements for machine learning software and its evaluation," *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2020-Augus, pp. 260–270, 2020.

[14] R. Zhang, A. Albrecht, J. Kausch, H. J. Putzer, T. Geipel, and P. Halady, "DDE process: A requirements engineering approach for machine learning in automated driving," *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 269–279, 2021.

[15] P. Höfner, R. Khedri, and B. Möller, "An algebra of product families," *Software and Systems Modeling*, vol. 10, no. 2, pp. 161–182, 2011.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. March, pp. 11 618–11 628, 2020.

Requirements debt: causes, consequences, and mitigating practices

Viviane Duarte Bonfim
Community University of Region from Chapecó
Chapecó, Brazil
Federal University of Santa Catarina
Florianópolis, Brazil
viviane.duarte@posgrad.ufsc.br

Fabiane Barreto Vavassori Benitti
Federal University of Santa Catarina
Florianópolis, Brazil
fabiane.benitti@ufsc.br

Abstract—Background: Agile development was an important initiative that changed the traditional way of developing software into an agile development process. Action is more important than the process in the agile world, and requirements and documentation do more harm than good. However, when developing requirement engineering activities inadequately, they motivate the emergence of problems that directly affect the development, which can incur requirements debt. Aim: This study investigates the causes of requirements debt that incur requirements debt and actions that can minimize and or avoid them inside the context of agile software development. Method: To fulfill the objective, we performed qualitative research, supported by data analysis suggested by Grounded Theory, with 19 subjects in 13 agile organizations at a national and international level in different segments. Results: At the end of this study, we proposed a theoretical model containing the requirements debt causes and their effects and practices that might mitigate them, and the relation between these three factors.

Keywords - Agile Requirements Engineering; Agile Software development; Requirements Debt.

I. INTRODUCTION

The current software development scenario is characterized by the broad adoption of agile methodologies [1] because the Agile Development Software (ADS) is, more and more, gaining space due to its crescent popularity and the possibility of quick and continuous deliveries [2]. It speeds up the development and adapting to the changes along the developing process. These changes suggest a flexible approach to software development, including Requirements Engineering (RE) [3].

Traditionally, the RE activities are developed separately from the development and design process and are documented in specific artifacts, promoting a formalization during this process [4]. In agile Requirements Engineering, the requirements are defined gradually along with the interaction between stakeholders and the developing team, without meeting the same formalization and, therefore, are not always adequately documented [4] in contrast to what the RE recommends [5], promoting a lack of standardization in the activities that comprise it [6].

The absence of a good requirements process may cause the RE conduction steps to fail, generating consequences such as misunderstood, omitted, ill-defined, and poorly specified requirements, including technical debts. Cunningham originally proposed in 1992 the approach as a metaphor for the term Technical Debt (TD), referring to the coding practices intending to help developers monitor the immature software code. This metaphor is related to software failure, generally motivated by development shortcuts or to the commitments made by developers to meet an urgent demand, convenient in the short term. With time, this concept evolved to other development stages, manifesting itself at the Requirements Engineering stage, also known as Requirements Debt[1] [8]. The requirements debt corresponds to failures in the requirements specification, characterizing the distance between the desired specification of requirements and the actual implementation of these requirements in the system [9]. A study performed by Ernest [10] initially revealed the requirements debt concept frequently adopted by researchers that address technical debt. These debts are still poorly understood by organizations, thus hindering the perception of their causes and their consequences. Hence, it becomes complex to prevent and treat them [11].

This research investigates the causes of the generated requirements debt, their consequences, and their mitigation actions. This study aims to allow agile organizations to understand better the scenario involving requirements debt and, in this way, mitigate them, improving their practices, aiming to prevent these debts and reduce the cost of their payment [12].

This article presents in section II the related works and, in section III, the research method adopted. Section IV presents the proposed theoretical model. Section V describes the results that supported the research. Section VI presents the threats to the validity of the research. Section VII addresses the conclusions, limitations of the study, and suggestions for future work.

DOI: 10.18293/SEKE2022-114

[1] The term initially proposed by [10] is characterized by technical debt in requirements, but recent studies address it as requirements debt [13], which this article adopts.

II. RELATED WORK

In the literature, several works involve research to investigate the conceptualization of technical debts in general, such as the work prepared by Freire et al. [12]. The researchers used data from the InsighTD project, which includes a set of surveys aimed at studying technical debts in Software Engineering, to investigate preventive actions that, if adopted, can curb the occurrence of technical debts and the difficulties in adopting these actions. The study proposed by Ramač et al. [14] demonstrates the understanding of the TD concept, together with data characterizing the causes and effects of TD in the information technology (IT) industry located in Serbia, obtained from 93 professionals.

However, few studies have investigated the requirements debt, such as the work developed by Lenarduzzi and Fucci [13]. The research carried out by Lenarduzzi and Fucci [13] presented a definition of requirements debt (ReD) that includes the debt incurred during the identification, formalization, and implementation of requirements. Lenarduzzi and Fucci [13] proposed three types of requirements debt: Type 0: Incomplete user needs; ReD Type 1: Requirement smells; ReD Type 2: Incompatible implementation. The authors suggested concepts and strategies for detecting, quantifying, and reimbursing each type.

The research developed by [13] is the only literature found that explores requirements debt and was peer-reviewed. However, they do not examine the evidence that causes the requirements debt, their effects, and strategies to prevent them. In this sense, no studies aim to investigate and identify the causes of requirements debt and their consequences and the practices that can mitigate these debts, specifically in the context of agile development, evidencing the need to carry out studies in this area.

III. RESEARCH METHOD

For the development of this research, an empirical study was carried out, following a qualitative approach guided by the Grounded Theory (GT) data analysis techniques supported by Charmaz's perspective [15]. Fig. 1 displays the flow of development of the research.

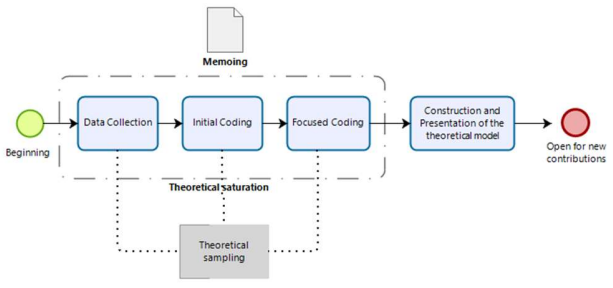


Figure 1. The process carried out during the research

The choice for GT is justified by its acceptance in the area of Software Engineering [16], as it facilitates the investigation of social and human aspects [17], and even being used to support data analysis, it results in consistent and valuable explanations about the phenomenon found [18], motivating the choice for this approach.

A. *Research Question*

This research aims to identify the causes that generate requirements debt and their effects, aiming at the activities of Requirements Engineering in agile organizations and actions that can mitigate the causes of these debts.

In this way, after the conclusion of the GT, it is possible to answer the following research questions:

RQ 1. What are the causes of requirements debt found in agile organizations?

RQ 1.1 What are the consequences of requirements debt?

RQ 2. What practices do agile organizations employ that can minimize requirements debt?

B. *Data Collection*

The analyzed data comes from the reports of 19 professionals in 13 agile organizations from 5 different countries (Brazil, France, Portugal, United Arab Emirates, and Belgium).

The research participants were selected from the researchers' contact networks or by indications from the companies' employees. The participants were contacted via e-mail, considering different profiles of employees, such as area of activity and the segment of companies, demonstrating through this diversity that theoretical sampling was achieved in the study, as shown in Table I. After the company and employees agreed to participate in the research, both filled out the consent form[2], and data collection took place.

It is relevant to mention that the participants did not need to be familiar with the term technical debt because if this aspect occurred during the interview, the researcher had the means to contextualize it to the participants.

The two instruments used for data collection were online interviews and a questionnaire. The chosen interview is semi-structured [19], as it has a previously defined script to help the interviewer in its conduction, yet, it is also supported by open questions [20]. The questionnaire[3] preparation considered the questions adopted in the interview. It was available through a form, and a single participant answered it. The participant belonged to a company located in Portugal. Such was the chosen process due to its availability and preference.

The interviews and questionnaire addressed aspects related to Requirements Engineering and Technical Debt in agile organizations and were recorded, with the participants' permission, followed by their transcripts, totaling 171 pages and

[2] Free and Clarified Consent https://forms.gle/uYFaoptYoAopKeuX6.

[3] Questionnaire: https://forms.gle/nRoMnSGBn1ijszQ8A.

approximately ninety thousand words between interviews and questionnaire.

It is noteworthy that the data collection and analysis ended only when the research reached theoretical saturation (when no new data emerged to add to the study [15]).

TABLE I. PROFESSIONALS PARTICIPATING IN THE RESEARCH

Participant profile			
Identification	**Role**	**Experience**	**Occupation area**
PA1	Project Manager	Experienced	Software or Internet Provider Management
PA2	Requirements Engineer	Experienced	Software or Internet Provider Management
PB1	Project Manager	Beginner	Solutions for Logistics and transportation
PB2	Project Manager	Beginner	Solutions for Logistics and transportation
PC1	Scrum Master	Experienced	Solutions for Logistics and transportation
PC2	Business Analyst	Beginner	Solutions for Logistics and transportation
PD1	Process Manager	Experienced	Hotel, restaurants and gas stations management
PD2	Project Manager	Experienced	Hotel, restaurants and gas stations management
PD3	Project Manager	Experienced	Hotel, restaurants and gas stations management
PE1	Requirements Engineer	Experienced	Digital transformation
PE2	Business Analyst	Experienced	Digital transformation
PE3	Requirements Engineer	Beginner	Digital transformation
PF1	Project Manager	Experienced	Solutions for Ministry of Health
PF2	Requirements Engineer	Experienced	Solutions for Ministry of Health
PG	Product Manager	Experienced	Technology solutions for the automotive sector
PH	Software Manager	Experienced	Uninformed
PI	Agile Coach	Experienced	Financial Institutions Products
PJ	Software Engineer	Experienced	Technology for the public sector
PK	Technical Leader	Experienced	Solutions for Airline
PL	Technical Leader	Experienced	Solutions for Public Sector
PM	CEO	Experienced	Solutions for Information security

C. *Data Analysis*

In the data analysis, coding techniques were used based on the Grounded Theory (GT) approach, supported by Charmaz [16], consisting of two steps for data coding: initial and focused. The GT assisted in the coding and data interpretation, and the MaxQda tool (https://www.maxqda.com/) supported performing the data analysis.

1) *Initial Coding*

Initial coding is the first step of data analysis following the constructivist approach [21], in which data are carefully examined, with line-by-line or segmental analysis of all transcribed material resulting from data collection [15]. With each interview or questionnaire, the transcription of the new data was coded and constantly compared with existing data. Table II presents segments giving rise to the different codes created, as reported by the research participants. We presented just a few examples of coded segments to support and represent the initial coding process.

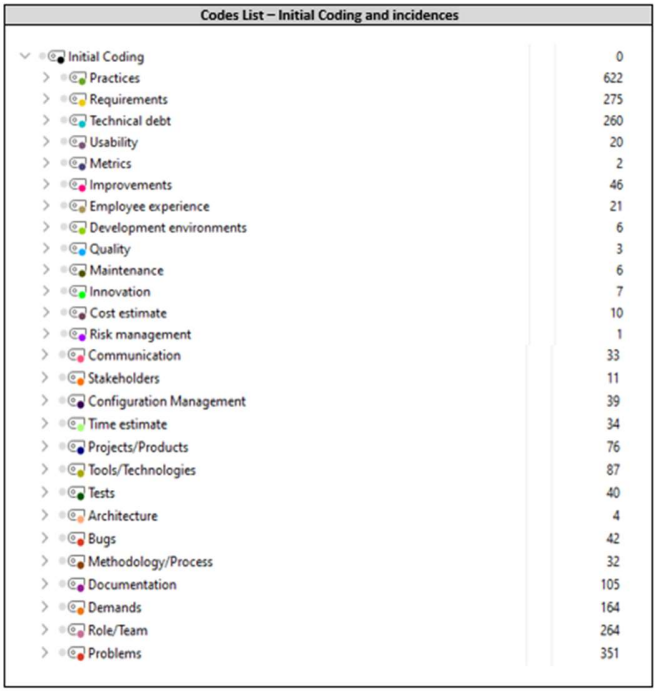
TABLE II. EXAMPLES OF PARTICIPANT REPORTS

Initial Coding	Participant Interview Segments	Participant
Requirements	*"This type of situation of requirements lacking generates a lack of adherence in an approved project and its development that we will have to return later to correct it, demand rework, generate DT, and create a complexity above what it should be".*	PC1
Technical debt	*"When business TDs happen, I believe they occur because of a lack of understanding or poor initial alignment, which was not mapped very well when the request arrived and was seen after a delivery that should have included it".*	PE1
Practices	*"We make references based on tags: e.g., workspace, documents, and notification. We are then able to connect everything, like a matrix".*	PJ
Problems	*"This process of talking to the customer to understand the requirements is not going well because it is tricky. Let's put it this way: this is a project thing. We don't have a person responsible for a feature (difficulty communicating with the customer)".*	PF1
Documentation	*"If we look at the software engineering content, this vision document is a requirement that will have functional and non-functional requirements. We call it a vision document. What for some people, I think would be that first, initial document, for us, it is inverted. Our vision document is already more detailed".*	PI

When a particular segment was selected, some related concept was verified. The new segment was associated with the existing code according to its similarity, proximity, and relationship to the created code. If it did not relate to an existing one, the analyzed segment resulted in a new code—all codes aimed at the software development process and related to Requirements Engineering and Technical Debt.

After several sessions of iteration and comparison between the data, at the end of the initial coding, 2599 coded segments were obtained, extracted from the interviews and questionnaire, and grouped into 27 codes, shown in Table III. Next to each code is its incidence (number of coded segments related to a given code).

TABLE III. CODES LIST



2) *Focused Coding*

According to the GT approach used, focused coding is the second stage of the coding process. In this step, the researchers analyzed the most relevant initial codes and organized them into subcategories and provisional categories that originated the final categories after refinement sessions. The grouping into categories and subcategories occurred according to identified similarities and differences [15], mapped and refined during comparative cycles in the data analysis, establishing connections between the categories and subcategories distributed in the following contexts: Technical Debt, Requirements Engineering, Agile Methodologies of development [22].

As a result, the researchers identified the main category of the research, represented by "Factors that impact Requirements Engineering" and its three final categories: "Causes that Generate Requirements Debt"; "Consequences that Characterize Requirements Debt"; and "Practices that can

reduce and/or address requirements debt." The categories have a set of subcategories that reflect the evidence of each of them. Tables IV, V, and VI show the subcategories. We chose this color model to match the theoretical model's subtitles in section IV. This evidence summarizes the reports verbalized by the participants during the data collection sessions and the interpretations made by the researchers and demonstrates perceptions conditioned to causes, consequences, and practices.

TABLE IV. "CAUSES THAT GENERATE REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Causes that generate requirements debts	Failure to receive requirements - Requirements debts Elicitation	- Failing to clarify the demand initially received from the customer
	Failed to refine requirements requisitos_TD_Requirements Analysis	- Requirements absence
		- Delaying the development of some demands or requirements
	Inconsistency in requirements specification	- Developing without specifying the requirements (generates rework)
		- Requirements specification failure
	Not validating before development_TD_Validation	- Not validating before developing (it was not as expected)
	Absence of requirements monitoring TD_Requirements Management	- Having backlog too extensive that are difficult manage (implicates time)
		- Lack of requirements monitoring/traceability

TABLE V. "CONSEQUENCES THAT CHARACTERIZE REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Consequences that characterize requirements debts	Functional and non-functional requirements (FR/FN) Inconsistency Requirements Analysis TD	- Requirements inconsistency or absence (FR/NFR)
	FR/NFR Not met due to a bad specification _TD_RE	- Lack of requirements due to a bad specification
		- Missing features and only realized customer´s feedback post-delivery
	Requirements not met realized in the delivery_TD_Validation	- Requirements not met and realized post- delivery
	Not everything that was requested was delivered _TD_Requirements Management	- Not everything is developed due to backlog accumulation/excess over time
		- Not everything that was requested was delivered (lack of requirements traceability)
		- Requirements/demands are dropped due to scope change

TABLE VI. "PRACTICES THAT CAN REDUCE AND/OR ADDRESS REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Practices that help reducing and/or prevent requirements debts	Reducing requirements debts	- Assessing impact of TD - usually pays debt in the next sprint
	Meeting requirements elicitation	- Recording all requirements, demands, and stories in the backlog
		- Building process flow or BDD to understand the demand
		-Performing requirements gathering through multidisciplinary workshops
	Helping requirements analysis	- Performing the requirements or demands refinement (sprint release)
		- Defining and validating what will be prioritized
		- Performing ceremonies (initials, sprint planning, sprint review)
		- Registering and classifying FR and NFR – using checklist for NFR
		- Following a requirements process
	Supporting the implementation of requirements specification	- Keeping and describing the requirements specification document clearly
		- Developing prototypes as part of the requirements specification document
		- Drawing up sequence and use case diagram and database model
	Implementing requirements validation	-Validating prototypes after requirements specification with customer
		-Performing experimentation, prototypes, MVP or customer's research
	Assisting in requirements management	- Managing the requirements, deliveries, backlog delays in progress
		- Performing requirements traceability

The focused coding, when completed, made it possible to build the theoretical model that underlies the entire study through the inductive/deductive process.

IV. THEORETICAL MODEL

The GT's coding theory supports the theoretical model presented in Fig. 2 and portrays the result of data analysis. During the elaboration of the theoretical model, it considered some findings that emerged from the data, supported by the analytical resources of the MaxQda tool.

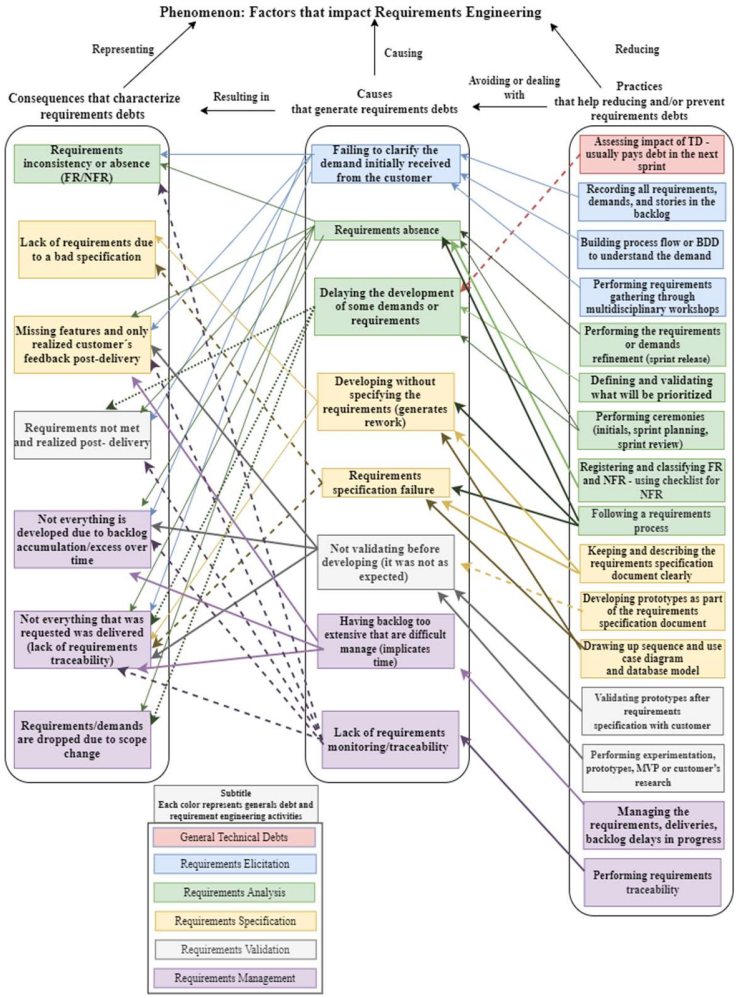


Figure 2. Proposed Theoretical Model

Three factors (categories) that impact Requirements Engineering characterize the model. The categories appear in three columns: Consequences highlighting the requirements debt themselves in the first column. In the central column are the causes for requirements debt, and in the third column, the practices that can minimize the causes and, consequently, the requirements debts.

The arrows employed define the relationship between these factors, indicating the causes that incur requirements debt and which practices can minimize or avoid them and consequently the debts. The option for the different styles and colors of the arrows intends to facilitate the understanding and legibility of the theoretical model.

A. *Data Validation*

The research data validation occurred with the participants by completing a form[4] after elaborating the theoretical model. The issues involved in the form are associated with the three factors that impact Requirements Engineering: Causes of requirements debt, their consequences, and the practices that can mitigate these debts. Of the 19 research participants, 10 participated in data validation[5], as some had left the companies they worked for at the time of data collection, and the other participants did not complete the validation form.

V. RESULTS

According to the model proposed in section IV, we identified eight leading causes that generate requirements debt, seven consequences that characterize the emergence of debts, and 16 possibilities of practices that can prevent and or treat these causes and consequently curb requirements debt.

After concluding the research, we believe that the theoretical model answered the research questions, establishing a connection between the categories, subcategories, and evidence resulting from the data analysis, unifying them, allowing a better understanding and identification of the causes that incur requirements debt and alternatives to minimize them.

The results obtained can help companies understand the causes of debt requirements, their effects, and what actions they can take to mitigate such debts. In the following sections, the participants' reports exemplify at least one cause, one consequence, and one practice due to space restrictions.

A. ***RQ 1. What are the causes of requirements debts found in agile organizations?***

Among the observed causes for requirement technical debt, the most reported cause was "Absence of requirements", evidenced by 13 participants. According to the mentions, the "absence of requirements" is a cause of requirements debt, perceived after delivery. Such debts point out that during the requirements analysis stage, the teams should refine a particular request to prevent identifying the absences only after delivery, as reported by PC1. According to the interviewees, the lack of understanding or initial alignment may reflex this cause.

"The absence of requirements causes a lack of adherence to the developed software. When this occurs, it is necessary to return to the requirements process to correct them, update them, and the following phases of the development process. This absence generates rework and additional cost, making the software development process more complex"[6]. – PC1, Scrum Master.

B. ***RQ 1.1. What are the consequences of Requirements Debt?***

We considered the consequence most frequently mentioned by the participants. The main one identified: "They do not develop everything – Accumulation or excess of the backlog over time," with 10 citations. The consequence mentioned makes management difficult, compromising the requested requirements, and often, no practice is adopted to facilitate monitoring. According to research participants, this accumulation occurs due to changes in priorities, as highlighted by PJ, since, with the backlog growing disproportionately, some requirements may be disregarded and never developed.

"We did not respond to everything that was requested. Many things that stay in the backlog are due to other requests, and they are priorities related to those in the Backlog". - PJ, Software Engineer.

C. ***RQ 2. What practices do agile organizations employ that can minimize requirements debt?***

Among the practices that can mitigate the requirements debt, we present the most expressive for the study, as reported by the participants: "Managing the Requirements, deliveries, delays, backlog, and progress," revealed by 15 participants. As highlighted in the interviews, the practice of "Managing the Requirements, deliveries, delays, backlog, and progress" allows controlling and monitoring the progress of requirements through their status, such as if they are already met, if they need development, and if they are late. It is also possible to track what is in the backlog, how long a particular demand or requirement remains there, and the requirements awaiting development. Some companies mentioned that to manage their requirements, they use specific tools because it allows them to explore the visualization of the status of each functionality/requirement through representative graphics, as mentioned by PJ.

"An Agile Board monitors all the steps: What is in the backlog, what is under analysis, awaiting development, in development, awaiting review, in review, awaiting test, in test, ready. Each of the requirements is in one of these blocks. In addition, we have the Burndown where we verify and follow up the deliveries concerning the sprint time". Agile Board integrates with Redmine. - PJ, Software Engineer.

VI. THREATS TO VALIDITY

There were mechanisms adopted in this study to mitigate some threats, highlighting some points described below. Focusing on construction validity, during the development of the study, we sought to explore the data collection instruments (interview and questionnaire) with participants from different

[4]Example of data validation: https://www.survio.com/survey/d/P7H8V2F7G2T4F7A4E

[5] Data Validation Results: https://drive.google.com/file/d/14m_cvUfuto8hrWJ1AD5_o0bFs_Xj7kd8/view?usp=sharing

[6] It is noteworthy that in the GT, transcripts must occur in full according to the participant´s speech. However, to provide better compression, there were some adaptations without changing context due to the article´s language.

countries and segments to absorb a significant set of data until data saturation occurred. When a participant did not understand a term or question in the interviews, the researcher was available to clarify. The same happened for questions related to filling the questionnaire out.

Considering the internal validity, the researcher's interpretation during the data coding was possibly not as faithful to the portrayed data from the interviews and questionnaire as it should be. This interpretation could reflect on the study´s results, even when the researchers reanalyzed the data when any doubts would arise. To minimize possible limitations, the research participants validated the results of the data analysis. However, of the nineteen research participants, ten participated in the validation, which can also characterize a threat to validity. A systematic study is underway to mitigate the threat regarding data validation. Said study will support validation and potentialize the obtained in this research.

VII. CONCLUSIONS AND FUTURE WORK

This article presented a study on the state of practice regarding the Requirements Debt involving 19 participants from agile organizations in different segments located in 5 countries. The research contemplated a qualitative approach supported by Grounded Theory data analysis techniques.

This study made it possible to observe that organizations do not have mechanisms to identify and recognize their requirements debts, and as a consequence, they do not manage such debts, making their mitigation difficult. To help organizations minimize these weaknesses, the theoretical model we propose synthesizes the results of this research. The theoretical model demonstrates the relationship between the identified categories and evidence, allowing agile organizations to recognize which causes generate requirements debts and their incurred debts for a better understanding and ways to minimize them by adopting the recommended practices.

In future work, we intend to provide a library of practices through a platform, suggesting a set of practices that help agile organizations minimize or mitigate requirements debt, regardless of the adopted process. We also mean to provide ways for companies to recognize the causes of requirements debt and their already existing debts to facilitate the implementation of practices.

REFERENCES

[1] N. Rios, M. Mendonça and C. Seaman. “Causes and effects of the presence of technical debt in agile software projects”. Twenty-fifth Americas Conference on Information Systems, AMCIS, 2019.

[2] K. Elghariani and N. Kama. “Review on agile requirements engineering challenges”. 3rd International Conference on Computer and Information Sciences (ICCOINS), August. 2016. DOI: 10.1109/ICCOINS.2016.7783267.

[3] E. Schön, D. Winter, M. J. Escalona and J. Thomaschewski. “Key challenges in agile requirements engineering”. International Conference on Agile Software Development: Agile Processes in Software Engineering and Extreme Programming, pp. 37-51, April 2017.

[4] E. Bjarnason, K. Wnuk and B. Regnell. “A case study on benefits and side-effects of agile practices in large-scale requirements engineering.” AREW '11: Proceedings of the 1st Workshop on Agile Requirements Engineering, pp. 1-5, July 2011. DOI: https://doi.org/1.1145/2068783.2068786

[6] H. F. Soares, N. S. R. Alves, T. S. Mendes, M. Mendonça and R. O. Spinola. “Investigating the link between user stories and documentation debt on software projects”. International Conference on Information Technology - New Generations, April 2015. DOI: 10.1109/ITNG.2015.68.

[7] W. Cunningham. “The WyCash portfolio management system”. Proc. OOPSLA, October 1992. DOI: https://dl.acm.org/doi/pdf/10.1145/157710.157715

[8] P. Avgeriou, P. Kruchten, I. Ozkaya and C. Seaman. “Managing Technical Debt in Software Engineering”. Dagstuhl Seminar 16162, January 2016. DOI: 10.4230 / DagRep.6.4.110.

[9] Y. Guo, R. O. Spínola and C. Seaman. “Exploring the costs of technical debt management – a case study”. Empirical Software Engineering, vol. 21. ed. 1. pp. 159-182, 2016. DOI: 10.1007/s10664-014-9351-7.

[10] N. A Ernst. “On the role of requirements in understanding and managing technical debt”. Third International Workshop on Managing Technical Debt (MTD), June 2012. DOI: 10.1109/MTD.2012.6226002.

[11] W. N. Behutiye, P. Rodríguez, M. Oivo and A. Tosun. “Analyzing the concept of technical debt in the context of agile software development: A systematic literature review”. Information and Software Technology, vol. 82, pp. 139-158, February 2017. DOI: https://doi.org/10.1016/j.infsof.2016.10.004.

[12] S. Freire, N. Rios, M. Mendonça, D. Falessi, C. Seaman, C. Izurieta and R. O. Spínola. “Actions and impediments for technical debt prevention: results from a global family of industrial surveys”. SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, pp. 1548–1555, March 2020. DOI: https://doi.org/10.1145/3341105.3373912.

[13] V. Lenarduzzi and D. Fucci. “Towards a holistic definition of requirements debt”. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), vol 1, pp. 1-5, September 2019. DOI: 10.1109/ESEM.2019.8870159.

[14] R. Ramač, V. Mandić, N. Taušan, N. Rios, M. G. Mendonça, C. Seaman and R. Oliveira Spinola. “Common causes and effects of technical debt in Serbian IT: InsighTD survey replication”. 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), August 2020. DOI: 10.1109/SEAA51224.2020.00065.

[15] K. Charmaz. “Constructing Grounded Theory: A Practical Guide through Qualitative Analysis”. Sage Publications, 2006.

[16] K. Madampe, R. Hoda, J. Grundy and P. Singh. “Towards understanding technical responses to requirements changes in agile teams”. IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW), pp 153-156, June 2020. DOI: https://doi.org/10.1145/3387940.3392229.

[17] R. Hoda and J. Noble. “Becoming agile: A grounded theory of agile transitions in practice”. IEEE/ACM 39th International Conference on Software Engineering (ICSE), May, 2017. DOI: 10.1109/ICSE.2017.21.

[18] R Hoda. “Decoding Grounded Theory for Software Engineering”. IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), May 2021. DOI: 10.1109/ICSE-Companion52605.2021.00139.

[19] W. C. Adams. “Handbook of practical program evaluation: Conducting Semi-structured interviews”. 3. Ed, Chapter Sixteen, pp. 365-376, 2010. ISBN: 978-0-47052247-9.

[20] J. Melegati, A. Goldman, F. Kon and X. Wang. “A model of requirements engineering in software startups”. Information and Software Technology, vol. 109, pp. 92-107, 2019. DOI: https://doi.org/10.1016/j.infsof.2019.02.001.

[21] K. Charmaz. “Constructing Grounded Theory. 2nd., 2014.

[22] P. Bourque, R. E Fairley. “Swebook. Guide to the Software Engineering”. Version 3.0. IEEE Computer Society, 2014.

On the Implications of Human-Paper Interaction for Software Requirements Engineering Education

Pankaj Kamthan
Department of Computer Science and Software Engineering
Concordia University
Montreal, Canada
pankaj.kamthan@concordia.ca

***Abstract*—It is broadly accepted that requirements engineering is one of the most important phases of a software project, and requires tools to be effective. For a variety of reasons, paper as a tool has lasted for millennia and remains ubiquitous. This paper makes a case for a contextual, conscientious, and evidence-based use of paper in a competency-oriented approach to software requirements engineering education (REE). It argues that the prophecies for the obsolescence of paper are premature, there are unique benefits in the use of paper, and the decision to use paper should be based on [0, 1] rather than {0, 1}. In this regard, a need-centered conceptual model for human-paper interaction is proposed. The characteristics of paper that make it historically unique are reported and the affordances of paper relevant to REE are discussed. The REE-related activities that benefit from viewing paper as a boundary object and using different types of paper are highlighted and illustrated by means of examples. In advocating polyliteracy, the potential for a convergence of paper and digital media towards a harmonic coexistence is underscored.**

Keywords-active learning; affordance; conceptual modeling; design thinking; human-centered agile methodology; software psychology

I. INTRODUCTION

The significance of *software requirements engineering* (RE) [1, 2] is underscored by the fact that it is a phase in which the stakeholders exercise considerable control over the success of the software project, and the decisions made during this phase usually have a major, often irreversible, impact on the subsequent phases. In the past 50 years or so, RE has evolved from an almost exclusively technically-oriented endeavor addressing mathematical problems to a contextually-, anthropologically-, and socially-sensitive discipline tackling ill-structured problems, such as "wicked problems". This change invariably impacts how *software requirements engineering education* (REE) should be perceived, planned, and pursued [3, 4], what the expected role of a software requirements engineer needs to be [2], and what the desirable competencies of a software requirements engineer are to be [5, 6].

As with many other software processes, a proper enactment of a RE process usually and inevitably involves using tools. The selection, adoption, and use of RE tools should be based on evidence rather than exuberance, understanding that professional tools do not automatically or necessarily meet the criteria of educational tools, determination that return on investment (ROI) >> 0, and consideration of the long-term consequences of a selection. One such candidate tool is paper. In that regard, the purpose of this paper is to investigate the extent to which paper can be useful as a tool for certain common activities in REE [7], the properties of paper that enable them, and the underlying reasons for this phenomenon.

DOI reference number: 10.18293/SEKE2022-007

The rest of the paper is organized as follows. In Section II, necessary background is provided and related work is discussed. A theoretical and practical understanding of the use of paper in REE from the perspective of human interaction is explored at some depth in Section III. In Section IV, potential directions for future research are outlined. Finally, in Section V, concluding remarks and recommendations are given.

II. BACKGROUND AND RELATED WORK

A. Characteristics of Paper relevant to REE

There are several distinctive, organic, and anthropomorphic characteristics of paper, such as the following:

- **Breathability.** It can retain (pencil) lead or absorb (pen) ink for a long period [8].
- **Emotivity.** It can give rise to different emotions among its users [9]. For example, a paper book can be perceived as a "beautiful object" (https://beautifulbooks.info/), and can add to the décor of a domicile. Indeed, people can create an emotional attachment with the paper books they have owned or read. The emotion can manifest in one or more different ways as, for example, identified by the *Plutchik's Wheel of Emotions*. For example, looking forward to and acquiring a paper book can make people happy, and losing it can make them sad.
- **Identity.** It can have a unique *persona* depending on the properties attributed to it during production (such as caliper, grammage, permeability, size, texture, and so on), making it recognizable even to those with visual impairment. For example, a paper book could be spotted from a distance, say, when it is on a shelf or table.
- **Resiliency.** It can be used even if it somewhat loses its original shape, say, is slightly crumpled, smudged, or torn. In other words, its utility and usability vary on a *continuous* set rather than on a discrete (binary) set.
- **Tangibility.** It can be touched and felt, and has friction. This, apart from physiological and psychological implications, creates a sense of *ownership*, and with ownership come *responsibility* and repercussions. For

example, ruining or losing a sheet of paper has meaning (as, at the very least, there is no automatic backup copy). In other words, "you broke it, you own it".

- **Temporality.** It can give a sense of passage of time (say, through signs of aging, decay, and smell), similar to a living being. This can bring about affinity and nostalgia among its owners.
- **Versatility.** It is a *boundary object*, and as such can be cut, flipped, folded, orientated, spindled, or torn in a variety of ways and shapes to suit users' preferences. For example, multiple, small sheets of paper can be produced "on-the-fly" from a single, large sheet of paper, and, conversely, a single, large sheet of paper can be created by gluing or taping together multiple, small sheets of paper.

These characteristics are not only among the reasons for the persistence of paper over millennia, but also have implications towards REE (and beyond), as discussed later.

B. Paper and the History of Computing

The history of large-scale programming in the 1950s, and subsequently of software engineering in the 1960s and 1970s, is an indicator of how the paper types and the degree of paper uses in these disciplines evolved, namely from more in the product I/O and less during the process to more during the process and less in the product I/O, as illustrated in Fig. 1 using color for emphasis. This transition could be attributed to the advancement of technologies for and the reduction in the cost of necessary hardware and software for I/O, and the increased attention on the principles and practices involved in processing.

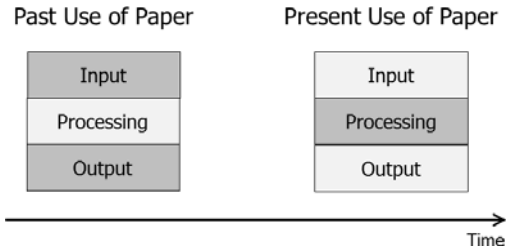


Figure 1. The uses over time of paper in computing.

The panorama of paper uses and paper types changed with the changes in the nature of computing, as some entered and the others exited. For example, the *punched card*, used for program input, is essentially obsolete because everything useful that was possible using it can be done otherwise, more effectively and efficiently. An almost similar argument could be made for *continuous paper* (such as the line printer paper), used for program output, as far as the consumption by public-at-large is concerned.

There have been calls since 2000s advocating the use of paper, albeit more so in human-computer interaction than in software engineering [10]. This situation, however, is changing as the two disciplines converge by necessity, such as seen by increasing human-centeredness of software development methodologies, in general, and agile methodologies, in particular [11]. For example, one of the values stated in the *Agile Manifesto*, namely "individuals and interactions over processes and tools", can be realized in practice if there is explicit attention on the *stakeholders needs* and there is inclusion of *lightweight tools*, such as paper. Indeed, this can be accomplished by integrating *design thinking* and *human factors design* in an agile methodology, such as *Scrum* [12]. For another example, the *Kanban Pizza Game* is played using pieces of paper representing the ingredients of a typical pizza [13]. However, there is much to be desired for making a case for paper in RE, in general, and REE, in particular, and that is one of the motivations of this paper.

C. Paper in Context from a Human Interaction Perspective

There have been a number of empirical studies over the years deliberating, evaluating, and reporting on relative merits of using paper and digital media for certain activities [8, 14]. (For the sake of this paper, *digital media* is some data presented using an application software, on a hardware device capable of digital computing, for the purpose of consumption by humans [8].) In that regard, it could be noted that paper and digital media appeal to different human senses [8, 15], reading on physical medium is different from that on digital medium [8, 16, 17, 18], and handwriting is different from typing [19].

An *affordance* is a property, or multiple properties, of an object that provides some indication to a user of how to interact with that object or with a feature of that object [15, 20]. Fig. 2 presents a Venn Diagram of two sets, one for the affordances of paper and the other for the affordances of digital media. In literature, the comparisons between paper and digital media are often restricted to comparison between C and B (= $(B - A) \cup C$), that is, anything that can be achieved with paper can also be achieved by digital media and digital media can achieve more, and do not consider $A - B$. C reflects early days of digital media when it tried to mimic and duplicate some of the affordances of paper [8].

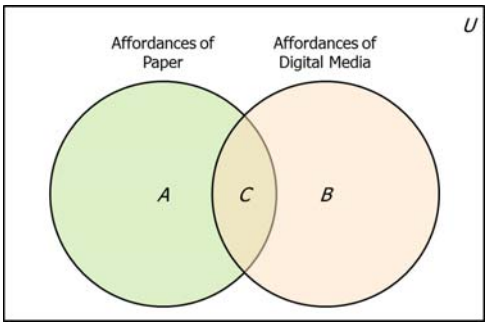


Figure 2. A comparison of affordances of paper and digital media.

It could be noted that, as far as affordances are concerned, there can be (1) perceived limitations of paper overcome by digital media, (2) perceived limitations of digital media overcome by paper, and (3) perceived limitations of paper not overcome by digital media *and* perceived limitations of digital media not overcome by paper, which is a proper subset of $U - (A \cup B)$. There are several examples of (1), such as automatic archivability, linkability, multimodality, retrievability, searchability, shareability, traceability, updatability, and so on, a discussion of which is beyond the scope of this paper. In addition, quality-in-use requirements are difficult to simulate properly on paper. For an example of (2), paper has a *single* level of abstraction, as implied by Fig. 3, while digital media

has *multiple* levels of abstraction (and, therefore, explicit dependencies), which has consequences for the usage of each. For an example of (3), requirements (such as those about credibility, maintainability, and reliability) that are a function of *duration* (that is, interval of, rather than point, in time) are difficult to simulate properly.

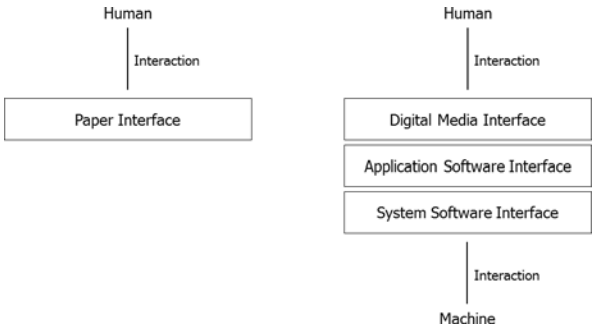


Figure 3. A comparison of levels of abstraction of paper and digital media.

III. A Human Interaction Perspective for Understanding the Use of Paper in REE

A. A Conceptual Model for Human-Paper Interaction and its Implications for REE

Fig. 4 shows a conceptual model in UML Class Diagram for human-paper interaction. The humans have needs, such as those highlighted by the higher levels of the *Maslow's Hierarchy of Needs*, some of which can be educational, as shown in Fig. 5. To satisfy those needs requires humans to draw upon (declarative and/or imperative) knowledge, which in case of REE is summarized in Fig. 6. This is then used to engage in one or more individual and/or social activities as, for example, explained by the *Activity Theory* [21, 22], in general, and the *Bloom's Taxonomy* [23], in particular. To make the communication or knowledge inherent to these activities explicit, they may need to be expressed in one or more artifacts, which could be made of some material, such as paper.

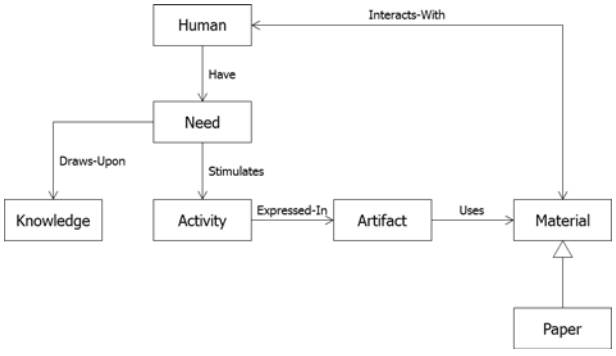


Figure 4. A conceptual model for human-paper interaction.

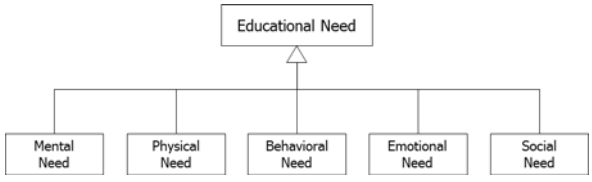


Figure 5. A hierarchy of educational needs.

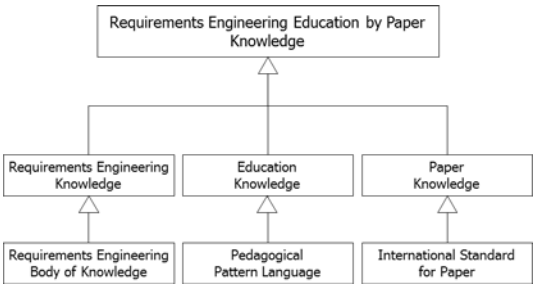


Figure 6. A hierarchy of REE by paper knowledge.

B. Paper Types Suitable for REE

There are many types of paper (https://papersizes.io/), of which some have been empirically proven to be useful in RE. The types of paper useful for REE can be either *generic* or *specific*, instances of which are shown in Fig. 7 and Fig. 8, respectively. The generic types of paper are broad in their applicability, and the mapping between the set of REE activities and the set of paper is many-to-many. The specific types of paper are narrow in their applicability, are available as device-specific templates, and the mapping between the set of REE activities and the set of paper is, essentially, one-to-one.

Figure 7. A collection of generic paper types relevant to REE.

Figure 8. A collection of specific paper types relevant to REE.

Fig. 9 highlights those properties of humans and paper that are relevant in human-paper interaction. For a given activity, selecting an appropriate type of paper is therefore important.

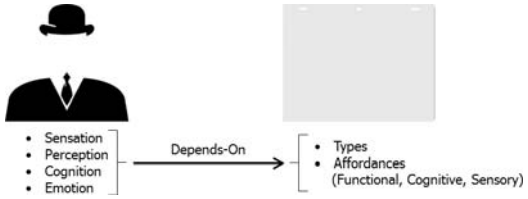


Figure 9. The human and paper properties in human-paper interaction.

C. A User Story Process Model for the Use of Paper

In general, a RE process is independent of the use of any particular tool, including paper. However, certain RE processes, especially those that are agile, human-centered, and informal, may be better suited to the use of paper than the others.

The user stories are one of the most common ways of expressing software requirements in human-centered agile methodologies [24]. Fig. 10 illustrates a user story process model that has been used for REE [25], the elements of which, namely *Express*, *Experiment*, and *Evaluate*, are extended, as appropriate, using the stages of design thinking, namely *Empathize*, *Define*, *Ideate*, *Prototype*, and *Test* [12], so that it becomes conducive to the use of paper. The symbol ► denotes the need for *convergent thinking*, while ◄ denotes the need for *divergent thinking*. The resulting model is aligned with the REE concepts given in the rest of Section III.

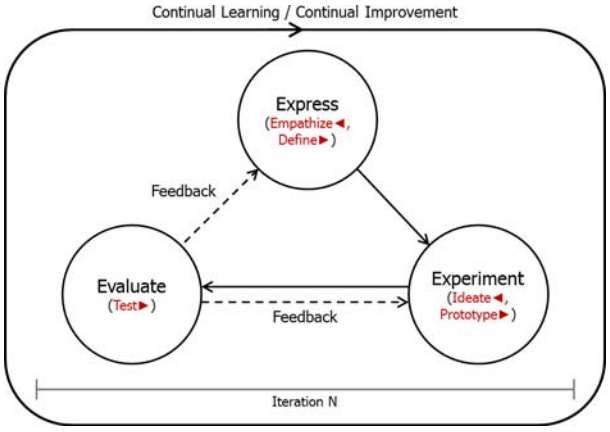


Figure 10. A user story process model conducive to the use of paper.

D. Implications of Paper for REE

The use of paper opens new vistas for REE, such as the following:

- **Silver Lining.** The perceived limitations of paper for REE (and beyond) can also happen to be its benefits. For example, the constraints of size (dimensions) of an index card compel a writer to be concise, which is recommended for user story and its acceptance criteria. The need to handwrite or draw for the others to be able to read and understand in a timely manner also obliges the writer to do so (or, if necessary, improve his or her handwriting and drawing skills accordingly), which are desirable lifelong skills for students.
- **RE Without Borders.** It is important for the students to understand that stakeholders of a software project usually include non-technical stakeholders who cannot be reasonably expected to be familiar with (or should be trained in the use of) digital technologies or tools used for software development. For example, non-technical stakeholders can include subject matter experts, business people, and potential end-users. The use of paper presents a low barrier of entry and fosters "democratic", "inclusive", and participatory design through face-to-face collaboration between technical and non-technical stakeholders.
- **Thinking and Doing in Tranquility.** The use of paper allows a person to dedicate time to think and concentrate on the matter at hand. (This is a consequence of Fig. 3.) There are no extra actions (no clicking, no loading-and-waiting, no panning, and no zooming) and no distractions (no advertisements, no clearing cache, no connectivity, no electrical power loss, no emission of heat or light, no error messages, no glare, no multitasking, no noise, no pop-up windows, no spellcheckers, no updates, and no viruses).
- **Creative Freedom.** The use of paper permits a person to draw freely, limited only by imagination. For example, there are no limits to the shapes and symbols such as those that could be used in a "boxes-and-lines" diagram, or, if necessary, invented "on-the-fly" such as while brainstorming or sketchnoting. There are also no a priori restrictions on where any text labels could be placed or how they may be spaced.
- **Preserving Memory of Mistakes.** In the use of paper, there is no "undo". The use of paper leaves physical reminders of any mistakes made by its user, however minor they may be, even if an eraser is used. These reminders can serve as evocative aides-mémoires of the quote "to err is human", RE smells or anti-patterns introduced and removed after 'iterative improvement' [26], and/or acknowledgement of 'lesson learned', hoping to not repeat the same or similar types of mistakes again. This—embracing and learning to live with one's mistakes—is crucial to lifelong learning of students.
- **Sustainability Lessons.** In software development, there can be different kinds of waste [27], including that of time and effort, such as due to rework. The provision of paper-based prototyping and feedback can help detect and correct certain types of errors early, thereby reducing rework later. The cost of paper and its impact towards environmental sustainability [28] can be a reminder to the students not to waste space and to use it conservatively, such as by using both of its sides. The waste of any kind should be discouraged and prevented, not least because it is one of the principles of *lean software development*. The movements such as the *World Paper Free Day*—an annual campaign that aims to reduce the amount of paper generated by people in their everyday work and personal life—should be encouraged and supported. The same applies to the *International E-Waste Day*. Indeed, these can be part of lifelong learning for students.
- **Preventative Approach to Development.** The use of paper enables getting the *right design* (validation) before getting the *design right* (verification). It is relatively easy and inexpensive to produce multiple design alternatives. (The need for delving into design in RE arises when undertaking a "wicked problem" where the act of finding a solution to the problem improves the understanding of the problem itself.) If a low-fidelity prototype is not accepted during user testing, chances are high that the end-product will not be accepted either.

- **Social Context.** The use of paper cultivates a natural environment for necessary socialization among stakeholders (including students) of a software project. For example, it can be used for meeting, discussing, and/or decorating in front of a Kanban board for showing the different *states* of work-in-progress in a hallway or in a classroom; planning poker using special-purpose playing cards for estimating user stories by stakeholders sitting around a table; and so on.

Incidentally, these observations contribute to REE by Paper Knowledge, as shown in Fig. 6.

E. REE Concepts in Practice on Paper

The REE concepts (interspersed and interrelated *activities* in a RE process and, possibly, *artifacts* resulting from those activities) are motivated by educational needs (as per Fig. 5). They could be divided into *primary concepts* (part of a RE process directly, and abstract) and *secondary concepts* (part of a RE process indirectly to support one or more primary concepts, and concrete).

The primary REE concepts include: active learning, collaborating, creating, discussing, empathizing, enjoyable learning, ensuring semiotic quality of software requirements (such as resolving ambiguities, inconsistencies, and indeterminacies), group learning, team building (norming stage to performing stage in the *Tuckman Model of Group Dynamics*), incrementing, iterating, negotiating, planning, problem solving, reading, thinking aloud, user testing, and writing.

Table 1 shows secondary REE concept(s), corresponding paper type(s), and supporting reference(s), wherever available. The symbol 'S' denotes the use by students in a course project.

TABLE I. SECONDARY REE CONCEPTS ON PAPER

REE Concept	Paper Type	Reference
Brainstorming, Computational Thinking, Doodling, Ideating, Mind Mapping, Sketchnoting	A, A1, Napkin	[7, 29, 30], S
Conceptual Modeling, Domain Understanding (Deciding Terms and Definitions for Software Project Glossary)	Sticky Note, A1	[7], S
Context Diagramming	A1	[31], S
Affinity Diagramming (Post-Requirements Elicitation Interview Analysis)	Sticky Note	[7, 32, 33], S
User Modeling (Eliciting Positive and Negative User Roles)	Sticky Note	[7], S
Empathy Mapping	A	S
Documenting User Stories and Acceptance Criteria	Index Card (Two Sides)	[7, 24], S
Estimating User Stories (Planning Poker)	Playing Card	[24, 34], S
Prioritizing User Stories	Index Card, Sticky Note	[24, 34], S
Customer Journey Mapping, User Story Mapping	A1	S
Information Architecting, Low-Fidelity Rapid Prototyping	Device Template, Grid, Kami, Ruled	[21, 33, 35, 36], S
Kanban Boarding	A1, Index Card, Sticky Note	[13]

IV. DIRECTIONS FOR FUTURE RESEARCH

There currently is no 'standard' RE pedagogical strategy, although there have been a number of notable initiatives over the years [2, 4]. There are also several possible paths through RE, which is why there are multiple possible courses on RE. For example, while one course may be oriented towards formal specifications for mission-critical systems, another may be oriented towards user stories for socio-technical systems. It would be useful to explore the variability in the use of paper with respect to different pedagogical strategies and different syllabuses in REE, and is therefore of research interest.

In Winter 2018 and Fall 2019, a survey on the use of paper in RE was conducted, the results of which were used in [7]. The respondees were graduate students in the course titled *SOEN 6481* (*Software Systems Requirements Specification*). The responses regarding preference for paper or digital media for RE was mixed. The comments from the students included: "*I have learned different uses of colored paper*", "*I have become better at reading others' handwritings*", and "*I was occupied enough with paper to not miss my smartphone!*". It would be useful to extend and repeat the survey, both during and after the COVID-19 pandemic, with both teachers and students of RE, and is therefore also of research interest.

V. CONCLUSION

The rich history and salient properties of paper make it uniquely suitable for a variety of REE-related activities, as this paper has shown. The circumstances presented by the COVID-19 pandemic have led to a notable decrease in face-to-face social interaction. These circumstances, invariably, have also necessitated, even accelerated, the use of digital media for some, a trend that may only continue, to which REE is not immune. This movement, however, should not come at a cost of use of paper. Indeed, the two can coexist [37, 38].

In conclusion, for teachers of RE there are following recommendations:

- **Recommendation 1: Careful Substitution.** There are no 'perfect' tools, tools are not substitutes for people and processes, and tools can aid, but are not a substitute for, thoughtfulness. Therefore, the students could be warned against the misconceptions and myths surrounding tools [8, 14], as well as drawbacks of shallow comparisons and impetuously-drawn sweeping conclusions regarding tools. The availability of digital LEGO® bricks (such as by using *LEGO® Digital Designer*) has not stopped the sale and use of physical LEGO® bricks. Similarly, the availability of interactive whiteboards has not made conventional blackboards useless or the students any smarter [39]. In accordance with building a pedagogical foundation for RE, it is only in students' interest to avoid being enamored by any particular tool and become *polyliterate*: learn to select multiple different tools, each based on its own merit, and learn to use them properly.
- **Recommendation 2: Spirited Cooperation.** The problems being addressed by software systems today have become so large and complex that they are not in the purview of any single individual if they are to be solved

within the given time and other constraints. Therefore, the students could be presented with opportunities not only to work collectively, but also to candidly review each other's work so that they can learn from their own mistakes as well as that of the others.

- **Recommendation 3: Rigorous Experimentation.** There are many possible views of software engineering, one of which is that it is a risky endeavor. Taking reasonable risks not only requires curiosity, but also courage to make mistakes early, and to learn and recover from them. Therefore, the students could be encouraged not only to seek the *known* iteratively and incrementally, but also the unknown and even the *unknowable* [40], all the while understanding the differences between them.

ACKNOWLEDGMENT

The author is grateful to CUPFA for a Professional Development Grant.

REFERENCES

[1] R. Siadati, P. Wernick, and V. Veneziano, Learning from History: The Case of Software Requirements Engineering. Requirements Engineering Magazine, September 25, 2019.

[2] M. Glinz, H. van Loenhoud, S. Staal, and S. Bühne, Handbook for the CPRE Foundation Level according to the IREB Standard: Education and Training for Certified Professional for Requirements Engineering (CPRE) Foundation Level, Version 1.0.0. International Requirements Engineering Board (IREB), November 2020.

[3] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, Requirements Engineering Education: A Systematic Mapping Study. Requirements Engineering, 20: 119-138, 2015.

[4] M. Daun, A. M. Grubb, and B. Tenbergen, A Survey of Instructional Approaches in the Requirements Engineering Education Literature. The Twenty Ninth IEEE International Requirements Engineering Conference (RE 2021), Notre Dame, USA, September 20-24, 2021.

[5] R. Klendauer, M. Berkovich, R. Gelvin, J. M. Leimeister, and H. Krcmar, Towards a Competency Model for Requirements Analysts. Information Systems Journal, 22: 475-503, 2012.

[6] S. Jantunen, R. Dumdum, and D. C. Gause, Towards New Requirements Engineering Competencies. The Twelfth International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE@ICSE 2019), Montreal, Canada, May 27, 2019.

[7] P. Kamthan and S. Hilal, On the Role of Paper in Agile and Active Requirements Engineering Education. The Forty Ninth ACM Technical Symposium on Computer Science Education (SIGCSE 2018), Baltimore, USA, February 21-24, 2018.

[8] H. Shibata and K. Omura, Why Digital Displays Cannot Replace Paper: The Cognitive Science of Media for Reading and Writing. Springer Nature, 2020.

[9] S. Fukuda, Emotional Engineering: Service Development. Springer-Verlag, 2011.

[10] D. Spinellis, On Paper. IEEE Software, 24(6): 24-25, 2007.

[11] T. S. da Silva, A. Martin, F. Maurer, and M. Silveira, User-Centered Design and Agile Methods: A Systematic Review. The 2011 Agile Conference (AGILE 2011), Salt Lake City, USA, August 7-13, 2011.

[12] A. R. Hoffmann, Sketching as Design Thinking. Routledge, 2020.

[13] M. Hammarberg and J. Sundén, Kanban in Action. Manning Publications, 2014.

[14] A. J. Sellen and R. H. R. Harper, The Myth of the Paperless Office. The MIT Press, 2002.

[15] D. A. Norman, The Psychology of Everyday Things. Basic Books, 1988.

[16] N. S. Baron, How We Read Now: Strategic Choices for Print, Screen, and Audio. Oxford University Press, 2021.

[17] Y. J. Jeong and G. Gweon, Advantages of Print Reading over Screen Reading: A Comparison of Visual Patterns, Reading Performance, and Reading Attitudes across Paper, Computers, and Tablets. International Journal of Human–Computer Interaction, 37(17): 1674-1684, 2021.

[18] M. Çınar, D. Doğan, and S. S. Seferoğlu, The Effects of Reading on Pixel vs. Paper: A Comparative Study. Behaviour and Information Technology, 40(3): 251-259, 2021.

[19] P. A. Mueller and D. M. Oppenheimer, The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking. Psychological Science, 25(6): 1159-1168, 2014.

[20] R. Hartson, Cognitive, Physical, Sensory and Functional Affordances in Interaction Design. Behaviour and Information Technology, 22(5): 315-338, 2003.

[21] C. Sibona, S. Pourreza, and S. Hill, Origami: An Active Learning Exercise for Scrum Project Management. Journal of Information Systems Education, 29(2): 105-116, 2018.

[22] O. Hazzan, T. Lapidot, and N. Ragonis, Guide to Teaching Computer Science: An Activity-Based Approach, Third Edition. Springer-Verlag, 2020.

[23] D. R. Krathwohl, A Revision of Bloom's Taxonomy: An Overview. Theory Into Practice, 41(4): 212-218, 2002.

[24] M. Cohn, User Stories Applied: For Agile Software Development. Addison-Wesley, 2004.

[25] P. Kamthan and N. Shahmir, A Framework for the Semiotic Quality of User Stories. The Twenty Seventh International Conference on Systems Engineering (ICSEng 2020), Virtual Event, USA, December 14-16, 2020.

[26] H. Femmer, D. M. Fernández, S. Wagner, and S. Eder, Rapid Quality Assurance with Requirements Smells. The Journal of Systems and Software, 123: 190-213, 2017.

[27] O. Shmueli and B. Ronen, Excessive Software Development: Practices and Penalties. International Journal of Project Management, 35: 13-27, 2017.

[28] Q. Kang, J. Lu, and J. Xu, Is E-Reading Environmentally More Sustainable than Conventional Reading? Evidence from a Systematic Literature Review. Library and Information Science Research, 43:1-11, 2021.

[29] D. Roam, The Back of the Napkin: Solving Problems and Selling Ideas with Pictures, Expanded Edition. Penguin, 2009.

[30] C. Wilson, Brainstorming and Beyond: A User-Centered Design Method. Morgan Kaufmann, 2013.

[31] K. Holtzblatt and H. Beyer, Contextual Design: Evolved. Morgan and Claypool, 2015.

[32] L. Ratcliffe and M. McNeill, Agile Experience Design: A Digital Designer's Guide to Agile, Lean, and Continuous. New Riders, 2012.

[33] B. T. Christensen, K. Halskov, and C. N. Klokmose, Sticky Creativity: Post-it® Note Cognition, Computers, and Design. Academic Press, 2020.

[34] M. Cohn, Agile Estimating and Planning. Prentice-Hall, 2005.

[35] C. Snyder, Paper Prototyping: The Fast and Easy Way to Define and Refine User Interfaces. Morgan Kaufmann, 2003.

[36] S. Greenberg, S. Carpendale, N. Marquardt, and B. Buxton, Sketching User Experiences: The Workbook. Morgan Kaufmann, 2012.

[37] J. Steimle, Pen-and-Paper User Interfaces: Integrating Printed and Digital Documents. Springer-Verlag, 2012.

[38] F. Han, Y. Cheng, M. Strachan, and X. Ma, Hybrid Paper-Digital Interfaces: A Systematic Literature Review. The 2021 Designing Interactive Systems Conference (DIS 2021), Virtual Event, USA, June 28-July 2, 2021.

[39] F. Gursula and G. B. Tozmaza, Which One Is Smarter? Teacher or Board. The Second World Conference on Educational Sciences (WCES 2010), Istanbul, Turkey, February 4-8, 2010.

[40] R. J. Barnes, D. C. Gause, and E. C. Way, Teaching the Unknown and the Unknowable in Requirements Engineering Education. The Third International Workshop on Requirements Engineering Education and Training (REET 2008), Barcelona, Spain, September 8, 2008.

Identifying Risks for Collaborative Systems during Requirements Engineering: An Ontology-Based Approach

Kirthy Kolluri, Robert Ahn, Julie Rauer, Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
{kirthy.kolluri, robert.sungsoo.ahn, julie.rauer, chung}@utdallas.edu

Tom Hill
Fellows Consulting Group
Dallas, TX, USA
{tom}@fellowsconsultinggroup.com

Abstract- A risk is an undesirable event that can result in mishaps if not identified early on during requirements engineering adequately. However, identifying risks can be challenging, and requirements engineers may not always be aware if risks are ignored. In this paper, we present *Murphy* – a framework for performing risk analysis. Murphy adopts the *Reference Model*, in which requirements are supposed to be met not by the projected software system behavior alone but through collaboration between the system and events occurring in its environment, hence the term *Collaborative System*. Murphy provides risk analysis facilities that include an activity-oriented ontology for carrying out risk analysis by systematically identifying risky activities in the system and in the environment, thereby obtaining a *Risk Analysis Graph (RAG)* and towards devising risk mitigation strategies later. In order to see both the strengths and weaknesses of Murphy, we experimented on developing a smartphone app involving a group of Ph.D. and senior-level graduate students – one group using Murphy and the other not using Murphy. Our observation, we feel, shows that the risks identified by the group using Murphy were able to identify more critical risks and those risks were comprehensive and relevant as. well. The results also showed that incorporating risk mitigation strategies for the risks identified can indeed help avoid them to some extent.

Keywords- ***Risk, Risk Identification, Ontology, Collaborative systems, Requirements Engineering, Reference Model (WRSPM Model)***

I. INTRODUCTION

Risk is a situation or event where humans themselves can be put at stake [13], is a phenomenon faced or caused by erroneous functionality/behavior of software, hardware, or human(s). Collaborative systems emphasize that requirements are satisfied by the collaboration between the user and the events in its environment. For example, in building our smartphone app, *Theia*[1], for helping blind people navigate indoors, it may not be too evident for the requirements engineer to identify that the "blind person may not be able to walk in a straight line." It can be challenging to determine the possibility of the smartphone app giving wrong instruction to the blind person, the camera not turning on even when the smartphone application is turned on, etc. Based on these examples, it is evident that risks may arise due to certain environmental events (domain) or erroneous system behavior.

When considering collaborative systems such as Theia, an agent (e.g., person, software, or hardware) must perform a set of activities to fulfill the requirement. Every action performed by the agent, or the software system is associated with one or more risks. What-if the smartphone app indicates the blind person to turn earlier or later after walking ten steps? Or what if the user ignores the instructions and fails to turn at the right spot? The requirements engineers and software developers should address these kinds of risks before developing the actual application. This practice would help plan risk minimization and mitigation strategies.

Some attempts have been made to perform risk analysis and mitigation during requirements analysis [1]. The idea here is to identify risks systematically, devise risk mitigation strategies and implement those strategies to help avoid some risks. But how do we systematically identify these risks without omissions and commissions and develop risk mitigation strategies?

This paper proposes *Murphy*, a framework for performing risk identification and analysis using an activity-oriented and ontology-based approach for collaborative systems. Our previous work [19], which extends the Reference Model [6, 7] with risks to obtain the Augmented Reference Model, is extended by adding more rules for systematic risk identification. We also introduce a highly activity-oriented ontology, a domain-specific ontology, and constraints on agent's actions which can be used to come up with risks when violated.

We carried out experimentation in two phases – *Phase 1* involved using the Theia app developed without using Murphy before its development. *Phase 2* involved using Theia app developed using Murphy framework. Through this experimentation, we have observed that significant risks, such as walking in a straight line, the user's finger being slippery to tap the screen, background noise, etc., can be overlooked. We also observed that the devising and developing risk mitigation strategies can indeed help avoid the occurrence of the risk to some extent.

DOI reference number: 10.18293/SEKE2022-169

[1] Theia is the Greek goddess of sight

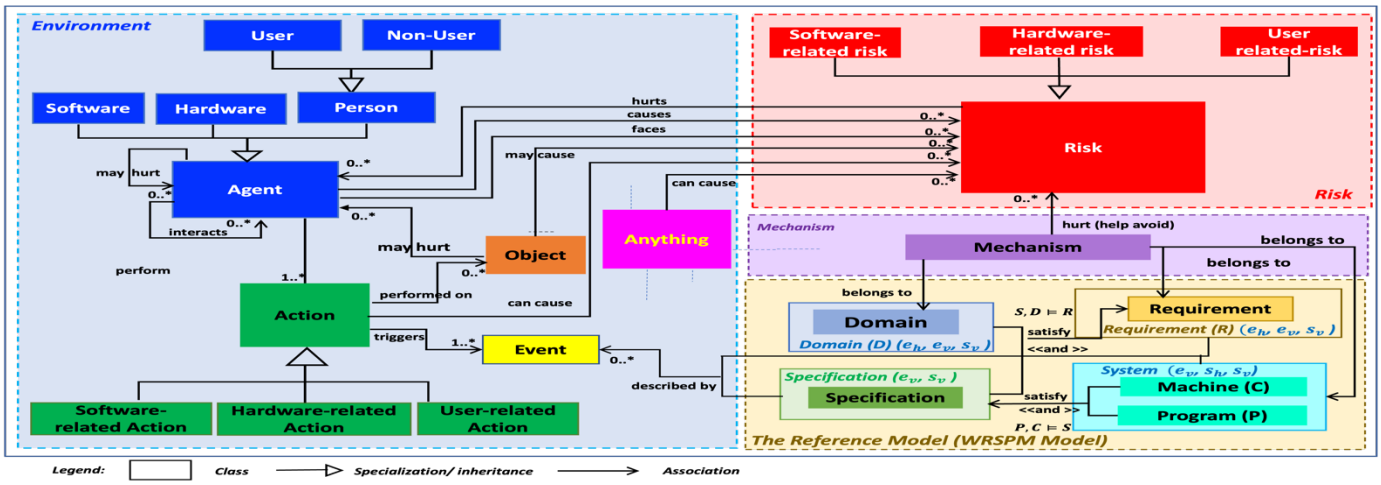


Figure 1: High-Level domain independent ontology of Murphy for risk identification and analysis

The main contributions of this paper are: proposing a risk analysis framework for capturing risks for various activit--ies performed by the Agent (Person, Software and Hardware) using multiple levels of ontologies during the requirement engineering phase. The framework suggests that risk mitigation strategies that must be implemented during the development of the application.

A scenario using the indoor navigation application, Theia, for helping blind people navigate indoors is used as the running example all through this paper, to evaluate the strengths and weaknesses of Murphy. Stevie is a blind student who wants to attend a class in room 3.415. He uses the smartphone application, Theia, to navigate from his current location to his class. He uses voice instruction to provide his destination to Theia.

Section II describes the related work. Section III describes Murphy framework using an ontology-based approach for risk identification. Section IV describes the experimentation and the observations, the discussion, and threats to validity. In the end, a summary of the paper is described, along with some future work in Section V.

II. Related Work

The distinctives of this paper include performing risk analysis for Collaborative systems in terms of the Augmented Reference Model, Risk Analysis Graph, and an Ontology-based approach, during requirements engineering.

Concerning risk analysis for collaborative systems, the CORAS framework [8], goal-risk framework [1, 10] and the Obstacle analysis technique [4, 5, 9, 11] are considered. All these frameworks consider non-functional requirements (goals) as their starting point and risks are eventually identified using different approaches, but we consider functional-requirements, specification, and domain assumptions as the starting point to perform risk identification and analysis.

In CORAS [8], risks are modeled and analyzed by asking questions, which are evaluated and treatments to those risks are identified. In the goal-risk framework [1, 10] goals, events and treatments are modeled in three layers, and they provide multi-object optimization; hence more queries related to risk are obtained qualitatively. Using the technique of obstacle analysis [4, 5, 9, 11], goals are decomposed into sub-goals, providing a set of rules including negation to identify the probability of risk occurrence quantitatively. Though our work has some similarities with [8, 1, 10, 4, 5, 9, 11] with regards to the approach of decomposing/refinements, and performing qualitative risk analysis, we complement the approaches used in [8, 4, 5, 9, 11] by performing systematic risk identification using an activity- oriented ontology. We also perform qualitative risk as discussed in [1] but we complement the analysis performed by using the Risk Analysis Graph, which provides a set of rules, using which the user can obtain and identify risks, prioritize them, and devise the corresponding risk mitigation strategies.

The Reference Model [6] emphasizes that the user requirements are satisfied not by the system alone but also by the system's collaboration with the events in its environment. Hence, we use the term Collaborative system for all the systems to which the Reference model is applicable. We adopt work involving the Reference Model [6, 7] and transform it into Augmented Reference Model [19].

In Requirements Engineering, a Fishbone diagram has been used to identify possible causes for a problem/risk [16]. This technique helps list out all the potential causes for a problem/risk. Fault Tree Analysis (FTA) is a top-down, deductive analysis that visually shows a failure path from top to bottom [17]. A Problem Inter-Dependency Graph (PIG) uses a (soft)problem technique to represent user's problem against the user's goals [18]. We build on the idea of finding the problems from Fishbone diagram [16], to use a top-down approach from FTA [17] and making refinements to identify risks from [18], but we also complement these techniques by proposing the use of a Risk Analysis Graph (RAG), that refines requirements, specification and domain using AND/OR refinements, and systematically generate risks using rules and using an activity-oriented ontology to identify the essential/critical risks.

The ontology of risk discussed in [12] explains its relationship with value. The ontology discussed in Requirements Modelling Language (RML) [14] address Requirements, Agent, action, etc. as the most important concepts. We adopt those basic building blocks from RML [14] and build the most essential part of our work - Risk, on top of it and tie the con-

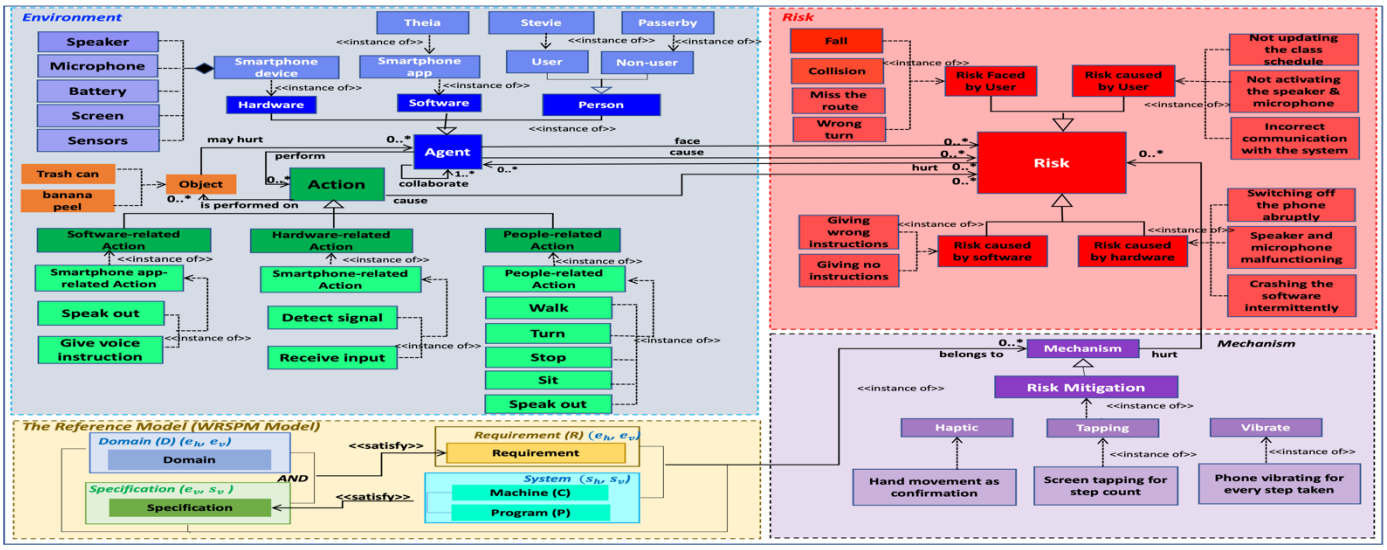


Figure 2: Domain specific activity-oriented ontology for smartphone application (Theia) domain

cept of Risk to Action and Agent. We also complement the approach discussed in [9, 12] by linking the ontology to actions and the risks that the user may face or cause.

III. MURPHY: AN ONTOLOGY-BASED FRAMEWORK FOR RISK IDENTIFICATION

The aim of our framework is to use an ontology-based approach to generate a Risk Analysis Graph using rules to identify risks, prioritize them and to find the appropriate risk mitigation techniques.

Ontology is the categories of essential individual concepts, relationship between the individual concepts and constraints on individual concepts and on the relationships between individual concepts. In this work, we present both high-level domain independent ontology and the domain specific class-level ontology with its diagrammatic convention as shown in Fig. 1, Fig. 2. We adopted several models and tightly integrated them into the ontology. Concepts for requirements engineering such as Requirement, Specification and Domain are adopted from Reference Model [6, 7] and the concepts such as Action (activities), Agent (entities) and Associations (assertions) from RML [14]. We further extend it by adding risk as shown in Fig. 1, which is one of the most essential concepts for this work alongside activities. Each concept is assigned a color and the same color is used for those concepts through the paper for traceability.

In detail, different types of *Agents* related to the domain such as Person, Software and Hardware are captured, the *Actions* pertaining to the agents such as Person action, Software action and Hardware action and the different types of *Risks* such as risks caused and faced by the person, risks caused by the software and risks caused by the hardware are captured. Capturing agents, their actions and the risks is vital because each agent performs a set of activities, and each activity is associated with a group of risks. In collaborative systems, agents interact and collaborate among themselves to fulfill the requirements. In addition, since the focus is on collaboration and collaborative approach, we capture all the critical concepts of the Reference Model [5] namely Requirement, Specification, Domain, Program and Machine in the ontology. Mechanism is captured to provide risk mitigation techniques for the risks discovered during risk identification and analysis process. Instances of the ontological concepts are represented in fig. 2 which is specific to Theia domain.

A. Omissions and Commissions*:* Using an ontology is one of the best ways to help ensure the completeness and comprehensiveness of the risk identification and analysis process. Identifying the most relevant and critical risks related to a domain help guarantee completeness, while identifying different kinds of risks is about comprehensiveness. Ontology helps avoid omissions and commissions of the most important/critical risks while performing risk analysis. Omissions are ignoring the most essential risks while commissions are identifying irrelevant or inconsistent risks. For e.g.,
R: When the person indicates a room number as the destination, the smartphone app shall ask the user to perform an action

By instantiating R using the class-level ontology shown in Fig. 2, it will be

r_1: When Stevie indicates his destination as room 3.415, Theia shall instruct Stevie to fly

The actions related to the agent (Person) as shown in Fig. 2 are walk, turn, stop, sit, speak out but we do not see fly as one of the actions related to Theia domain. Hence, this is an example of *commission*. Similarly, another instance for R can be:

r_2: *When Stevie indicates room 3.415 as his destination, Theia shall instruct Stevie to walk 10 steps forward.*

Stevie walking 10 steps forward can have so many risks associated with it. He may not walk forward at all but *backwards*, or he may not walk in a straight line but in a *zig-zag pattern*, or does not hold the camera facing forward but *downwards,* etc. All these are omission of substantial risks in relation to Theia domain. Stevie not hitting the brakes, Stevie not chang-

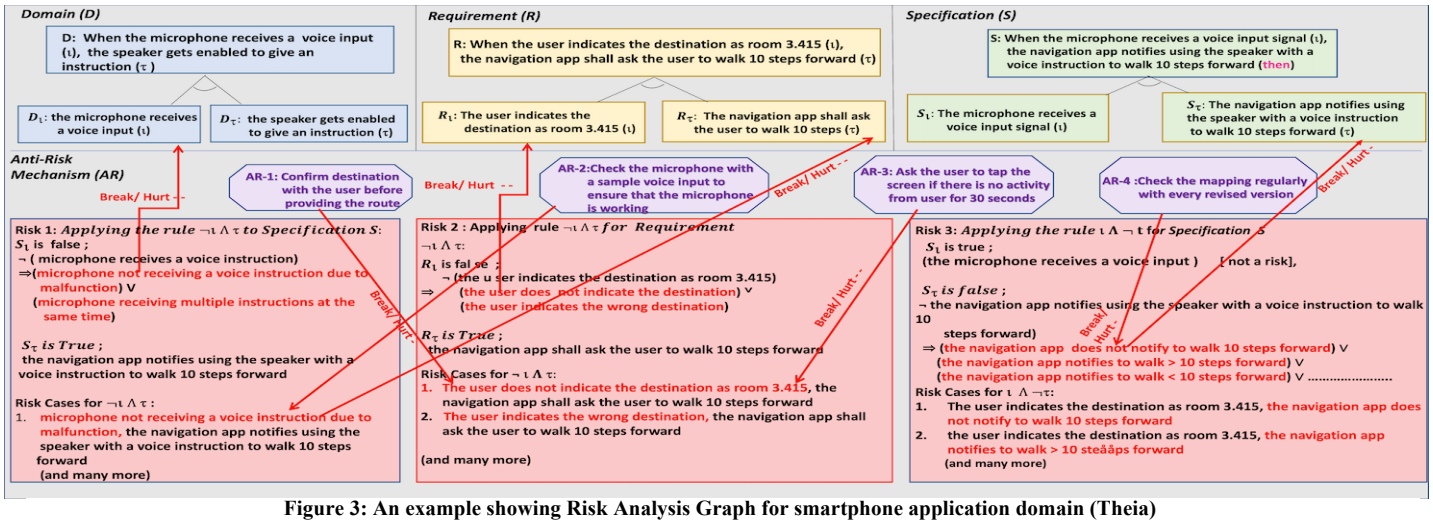


Figure 3: An example showing Risk Analysis Graph for smartphone application domain (Theia)

ing lanes, etc. are also risks but are not omissions related to Theia domain.

***B. Identifying risks using Constraints*:** Constraints are some restrictions that are placed on the ontological concepts and the relationships between them. Since, the ontology is highly activity-oriented, we place constraints and violation of these constraints is nothing but a risk. The constraints are specific to the domain and are related to the actions that the agent perform in the domain. For e.g., if we consider a constraint that the blind person must walk in a straight line when using Theia. The blind person walking in a zig-zag pattern can violate this constraint, which results in a risk.

***C. Generation of Risk Analysis Graph*:** In this paper, we generate Risk Analysis Graph, shown in Fig. 3, by the systematic generation of risks. For this, we use the Augmented Reference Model from our previous work [19] and extend it by adding multiple rules for extensive risk generation and devising risk mitigation strategies for the risks identified. As a part of the risk analysis process, we follow these steps to generate the RAG:

1) acquiring the requirement, specification, and domain,
2) decomposing the requirement, specification, and domain
3) applying rules to the decomposed requirement, specification, and domain to systematically obtain risks
4) prioritize the most important risks using ontology
5) devise risk mitigation strategies to the risks prioritized

Step 1 and 2: For this work, we assume that the requirement, specification, and domain are of the form $\iota \rightarrow \tau$. We use the antecedent (ι) part and the consequent (τ) part to identify risks by applying rules for systematic risk generation. We use a part of the initial process (Steps 1 and 2) discussed in our previous work [19] for generation of RAG.

Step 3: We have discussed some rules in our previous work [19], and we extend them by adding more rules for systematic risk generation. Rules are applied to the decomposed requirement, specification and domain that are obtained using the augmentation process explained in our previous work [19].

a) Rule 1: $\neg(\iota \rightarrow \tau)$ which is $\iota \wedge \neg\tau$

b) Rule 2: $\neg\iota \wedge \tau$: We consider the antecedent and the negation of the consequent joined by a logical AND

c) Rule 3: Negation of Contrapositive: Contrapositive is the reversal and negation of both i and t in $\iota \rightarrow \tau$. It is read as if not t then not i. We consider negation of the contrapositive [15], *represented as $\neg(\neg\tau \rightarrow \neg\iota)$*

In fig. 3, the first two red boxes show rule 2 in action and the last red box (towards right) shows rule 1 in action. We will discuss only rule 1 here (the last red box in fig 3) due to space limitation. Let us consider the specification S,

S: When the hardware receives a signal, the software notifies using the hardware to perform an action

To generate RAG, we perform step 1, i.e., acquiring s. We instantiate this S, using the ontology. By instantiating this specification S, we acquire:

s: When the microphone receives a voice input signal, Theia notifies using the speaker to walk 10 steps forward

We then perform step 2, i.e., decomposing s since we have an implies relation between ι and τ. After decomposing s, we obtain s_ι *and* s_τ.
s_ι: *the microphone receives a voice input signal*
s_τ: *Theia notifies using the speaker to walk 10 steps forward*

Now, we apply rule 1, $\iota \wedge \neg \tau$, to s_ι and s_τ. When rule 1 is applied to s, the consequent, s_τ, is negated since the consequent in $\iota \wedge \neg \tau$ has the negation.
The antecedent remains the same, and the relation between them is AND. By negating s_τ, we obtain:
(s_τ): ¬ (*Theia notifies using the speaker to walk 10 steps forward)*

There can be multiple risk cases associated with this negation.
(Theia does not notify to walk forward) OR
(Theia notifies to walk > 10 steps forward) OR
(Theia notifies to walk < 10 steps forward) OR
(Theia notifies to walk 10 steps forward and turn left) OR
(Theia notifies to walk < 10 steps forward and turn left)

When Theia must deliver a notification using the speaker, after calculating the route, there may be a set of risks that can be associated with a simple statement. To calculate the route and give an instruction, there is an action that the agents Theia (software) and speaker (hardware) must perform. As discussed earlier, each action that an agent performs, can be associated with one or more risks. Hence, the resulting risks could be Theia does not calculate the route and no instruction is given, or Theia calculates wrong route, etc. Alternatively, the route calculation by Theia may be perfect but the speaker may not give out the instruction. We identify templates using rules, implement these templates in our tool, to generate risks when a requirement, specification or domain statement is provided.

***Step 4*:** Risks obtained from step 3 are prioritized using the ontology. All the risks are compared against the set of risks listed in the domain specific, class-level ontology. The risks listed in the ontology are prioritized and the risks that are irrelevant (commissions) are ignored.

Step 5: *Risk-mitigation techniques* are devised based on the risks prioritized in step 4 which are shown in purple in fig 3. The break/hurt arrow represents that a risk mitigation technique hurts the risk, and it prevents that risk from happening. For e.g., if we prioritize the risk – the speaker may not give out the instruction since the speaker does not work, the risk mitigation technique that may help avoid that risk is to include a test voice/music clip which can be played by the user to make sure that the speaker is working before indicating the destination, etc.

D. Murphy Assistant tool: Murphy Assistant is a semi-automated Risk Analysis tool, where the user of the application has to setup the ontology before performing risk identification and analysis. For this process, we developed a windows application using the .NET framework. For storing all ontological concepts entered by the user, a Microsoft SQL Server Local Database is used. Murphy Assistant is a prototype tool which supports the concepts of Murphy framework. The refinement rules are provided as templates to this tool, and these templates are semantically bound. The underlying code can and the snapshots of the tool in action can be found at https://github.com/indoornavigation0/Murphy.git

IV. Experimentation

To validate our risk analysis and to identify the strengths and weaknesses of Murphy, we design an experiment to develop a smartphone app from the results obtained by performing risk analysis using Murphy.

***A. Experimental Setup*:** Murphy is intended to be used by requirements engineers and developers to perform risk analysis during the software development life cycle before the development of the application. We have conducted experiments, to validate Murphy with the help of a group of 25 PhD and 25 senior-level graduate students. All the students majored in computer science. Every student was provided with a version of Murphy installed on their computer. The students were given the requirement that we used as the running example, and tested many different requirements of their choice, chose the branches for which risk analysis should be performed, chose the rules to be applied to requirement, specification, or domain and when to stop the risk analysis (depth). After using Murphy, the students provided us with the list of risks and their feedback regarding the ease of use, accuracy of the automation and its usability, along with a list of risks identified.

A version of Theia has been developed using the list of risks and risk mitigation strategies provided by the students. One such mechanism has been implemented in Theia. We have conducted experimentation using Theia with 25 students.

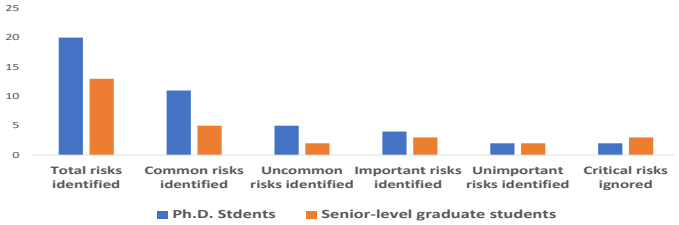


Figure 4: Different categories of risks identified using Murphy

***B. Analysis of the result*:** Performing risk analysis is a vital step before the development. Exclusion of RAG, as per our observation, we feel, shows that some risks are omitted. All the students were able to find common risks related to agents malfunctioning. We have observed various kinds of risks where the system did not behave the way it was supposed to. We also identified some risks where the system's functionality was aberrant. There were some risks which were not very relevant to the domain.

Students identified risks such as missing route, walking in the wrong direction, etc., critical risks such as falling, bumping into people, colliding against walls, etc., uncommon risks such as oil on the floor, banana peel on the way, water puddle on the floor, etc., unimportant risks such as warnings which ask the user to increase the volume, increase screen brightness, etc. The students have ignored some critical risks such as low battery indication, faulty voice input due to background noise, walking in a zig-zag fashion in a straight corridor, the blind person walking into a busy intersection, the user walking wrong number of steps, etc. These results are discussed in fig. 4, a bar graph which shows the number of risks in different categories identified by both the Ph.D. and Graduate-level students. Fig. 5 shows a bar graph, which compares the results of the risks faced/ignored while using the version of Theia developed without Murphy versus the version of Theia developed with Murphy.

In the version of Theia developed using results from Murphy, the system counts the step as the user walks, to help ensure safety of the user and to keep track of the steps walked by the user. This is the biggest difference between the app developed by using results from the Murphy versus not using the results from Murphy. Based on our observation, we feel that the students who used Theia (developed using Murphy's results) were able to walk very confidently since Theia was counting steps for them while navigating. Most of the students were able to make an accurate turn at the right spot, were able to keep track of their steps and were able to enjoy the process of navigation with ease. Overall, we feel that the results observed from these experiments show us that performing risk analysis during requirements engineering can help the end user avoid risks to some extent.

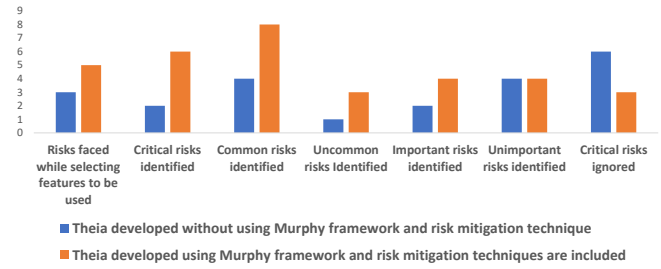


Figure 5: Results observed while using Theia developed without using Murphy vs using Murphy framework

C. Threats to Validity: We feel that our experiments have shown that there is a need to improve existing smartphone apps and devices for blind people, especially with features that ensure that the blind person is comfortable while navigating using the smartphone app, by building a tool for identifying risks systematically, devising anti-risk mechanisms and incorporating those results into the system before development. The Murphy Assistant tool needs improvements with more rules and templates to identify more complicated and uncommon risks. Since both Murphy Assistant and Theia are tested by students, and since the knowledge of the students is limited, the results may vary greatly compared to the app being tested by requirements engineers. The risks identified also varied greatly from what we anticipated since the use of the tool is based on individual knowledge and the way of performing analysis varies from person to person, therefore our results suffered. We are yet to test our Theia app with real blind people as we are yet to receive our IRB approval. We feel that testing with real blind people may give us an edge over blindfolded people, especially with identifying a variety of risks they face.

V. CONCLUSION

In this paper, we presented Murphy - a framework for performing risk identification and analysis using Augmented Reference Model - The Reference Model augmented with risks that was extended drawing to our previous work [19]. In this paper, we presented: 1. An activity-oriented ontology to perform risk analysis, 2. Risk Analysis Graph - for identifying and prioritizing and to devising risk mitigation techniques, 4. A tool, Murphy Assistant developed as a proof of concept, to generate RAGs for different requirements, specifications, and domains. 5. A reference application Theia is used to evaluate the strengths and weaknesses of Murphy. Based on the feedback from the students who used Murphy, we feel that its use during requirements engineering can indeed help increase the confidence of the engineers and developers in identifying some critical risks.

As future work, we plan to apply our approach to a wide variety of domains (e.g., autonomous vehicles domain) for performing risk analysis and providing risk mitigation strategies. We are developing a set of rules which aid in risk identification and analysis which goes beyond logic (simple negation). Experimentation of Theia with real blind subjects will be performed once we obtain the IRB approval. A step-by-step approach for engineers to develop and design their own graphically oriented Risk Analysis Graph's (RAGs) and identifying risks is underway as well. Finally, we plan to include safety and timeliness as a softgoal and extend our work using a goal-oriented approach.

REFERENCES

[1] Asnar, Y., Giorgini, P. Mylopoulos, J. Goal-driven risk assessment in requirement engineering. Requirements Eng 16, 101–116 (2011). https://doi.org/10.1007/s00766-010-0112-x

[2] Murphy's Law, https://en.wikipedia.org/wiki/Murphy's_law , Last accessed 2 February 2022

[3] Sharma, K., Kumar, P.V.. (2014). A method to risk analysis in requirement eng neering through optimized goal selection tropos goal layer. Journal of Theoretical and Applied Information Technology. 61. 270- 280.

[4] Cailliau, A. Lamsweerde, A. V.. "A probabilistic framework for goal-oriented risk analysis,"(2012). 20th IEEE International Requirements Engineering Conference (RE). Chicago, IL, 2012, pp. 201-210. doi: 10.1109/RE.2012.6345805.

[5] Lamsweerde, A. V.. "Risk-driven Engineering of Requirements for Dependable

[6] Gunter, C. A., Gunter, E. L., Jackson, M. Zave, P."A reference model Systems." En gineering Dependable Software Systems for requirements and specifications," in IEEE Software, vol. 17, no. 3, pp. 37-43, May-June 2000, doi: 10.1109/52.896248.

[7] Zave, P., Jackson, M.. (1997). Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. 6, 1 (Jan. 1997), 1–30. DOI:https://doi.org/10.1145/237432.237434

[8] Vraalsen F., den Braber F., Lund M.S., Stølen K. (2005) The CORAS Tool for Security Risk Analysis. In: Herrmann P., Issarny V., Shiu S (eds) Trust Manage ment. iTrust 2005. Lecture Notes in Computer Science, vol 3477. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11429760_30

[9] Lamsweerde, A.V.. (2013). Risk-driven engineering of requirements for dependable systems. 10.3233/978-1-61499-207-3-207.

[10] Mylopoulos, J. Castro, J.."Tropos: A Framework for Requirements- Driven Software Development," (2000).INFORMATION SYSTEMS ENGINEERING: STATE OF THE ART AND RESEARCH THEMES, pp. 261-273.

[11] Cailliau, A., van Lamsweerde, A. Assessing requirements-related risks through probabilistic goals and obstacles. Requirements Eng 18, 129–146 (2013). https://doi.org/10.1007/s00766-013-0168-5

[12] Sales T.P., Baião F., Guizzardi G., Almeida J.P.A., Guarino N., Mylopoulos (2018) The Common Ontology of Value and Risk. In: Trujil.lo J. et al. (eds) Conceptual Modeling. ER 2018. Lecture Notes in Computer Science, vol 11157. Springer, Cham. https://doi.org/10.1007/978-030-00847-5_11

[13] Rosa, E.. "Metatheoretical foundations for post-normal risk." Journal of Risk Research 1 (1998): 15-44.

[14] Greenspan, S., Mylopoulos, J. Borgida, A.. 1994. On formal Requirements modeling languages: RML revisited. In Proceedings of the 16th international conference on Software engineering (ICSE '94). IEEE Com. puter SocietyPress, Washington, DC, USA, 135–147.

[15] Contrapositive, https://en.wikipedia.org/wiki/Contraposition". Last accessed 29 September 2021

[16] Ishikawa, K.: Introduction to quality control. Productivity Press (1990)

[17] Vesely, B.: Fault tree analysis (fta): Concepts and applications. NASA HQ(2002)

[18] Supakkul, S., Chung, L.: Extending problem frames to deal with stake Holder problems: An agent-and goal-oriented approach. In: Proceedings of the 2009 ACM symposium on Applied Computing. (2009) 389-394

[19] K. Kolluri, R. Ahn, T. Hill, L. Chung, " Risk Analysis for Collaborative Systems during Requirements Engineering", " Proc., International Conference on Software Engineering & Knowledge Engineering (SEKE 2021). 2021, pp. 297-302.]

A Novel Approach to Maintain Traceability between Safety Requirements and Model Design

Qian Wang[†], Jing Liu[†*], John Zhang[‡*], Hui Dou[‡], Haiying Sun[†], HongTao Chen[‡], Xiaohong Chen[†], Jifeng He[†]
[†]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
[‡]Huawei Technology, Shanghai, China
[*]Corresponding authors: Jing Liu (Email: jliu@sei.ecnu.edu.cn), John Zhang (Email: john.zhangyh@huawei.com)

***Abstract*—One of the major challenges confronting System Modeling Language(SysML) is that it cannot always provide verifiable guarantees of formalization and rigorousness. To verify model designs, the research of transformation from SysML to ontology emerges because of ontology's formal standards and verifiability obtained by ontology reasoners. However, existing transformation approaches are mostly limited to a single view without traceability or lack a clear process so that it can't be automated. In this paper, we propose a novel approach to maintain precious traceability between requirements and model multi-views design based on ontology. In addition, our approach contains a normative process of ontology building in support of an automated implementation. We use this approach to obtain the ontology of a safety-critical system and carry out the ontology evaluation experiment, whose results demonstrate the feasibility and efficiency of our approach.**

***Index Terms*—Knowledge Engineering, Traceability, SysML, Description Logic, Ontology, Model Transformation**

I. Introduction

As a knowledge representation tool, System Modeling Language (SysML) has been used in many safety-critical systems. In the most common SysML usage mode, i.e., *SysML-as-Pretty-Pictures*, it uses various diagrams to express the requirements, structure, and behavior of the system. Unfortunately, this mode lacks complete formalization and rigorousness, and users pay relatively little attention to the well-formedness of SysML and its underlying simulatable and executable semantics [1]. Hence, the generated models are difficult to drive dynamic behavior simulations to confirm that the system behavior satisfies the safety requirements. The challenge will bring in more time and labor costs [2].

To address the aforementioned challenge, the research of transformation from SysML to ontology and verification based on ontology emerged. Ontology is extended based on description logic which has formal standards; therefore, ontology reasoners can find errors in ontology rules and facts while drawing inferences [4]. If a SysML model is transformed into an ontology with the same semantics, we can conduct inferences to verify the design of models.

In this paper, to address the problem, we propose a novel approach to maintain precious traceability between requirement and model design based on ontology. The establishment of maintaining traceability from risk to overall safety requirements, to safety-critical system component design, is of great significance for behavior simulations [7]. Specifically, we first construct the high-level ontology to formally represent concept elements and semantics of SysML using Web Ontology Language 2(OWL 2) [8]. Then, models' instances are added into the high-level ontology to generate low-level ontology, which can be applied to verify and query using well-developed OWL reasoners. Finally, we conduct experiments on a real-world safety-critical system. The experimental results show that the ontology generated by the proposed approach is of good overall quality, and there are no pitfalls that affect the consistency, reasoning, and applicability of ontology.

The contributions of this paper are threefold: (1)an innovative approach to maintaining traceability using a phased transformation, (2)a normative process of ontology building in support of an automated implementation, (3)an experimental evaluation of a real-world safety-critical system with 23 requirements showing feasibility and efficiency of our approach.

DOI reference number: 10.18293/SEKE2022-140

II. Transformation Approach

A. Ontology Generation

Transformation in this paper concentrates on SysML's four semantic dimensions: (1) requirements, (2) structure, (3) behavior, and (4) state. After the reference to [9] where shows a normative ontology building process to guarantee efficiency, the transformation consists of six steps. The first four steps create the upper-level ontology(written $\bar{O}$ for ease of understanding), and the last two steps derive the lower-level ontology(written o) of a specific model m from $\bar{O}$. The following are the detailed steps.

Step 1. Determine the domain and scope of the ontology. The way of starting the development of an ontology is to answer several questions:

- What is the domain? SysML's four semantic dimensions.
- For what we are going to use? Complete and correct expression of the domain's semantics, and then the users can query and infer about m from o.
- What types of information can the ontology provides? Models' basic and traceability information.
- Who are the ontology's users? Software engineers.

Step 2. Enumerate important terms. This step focuses on writing down important terms. The semantics of a dimension

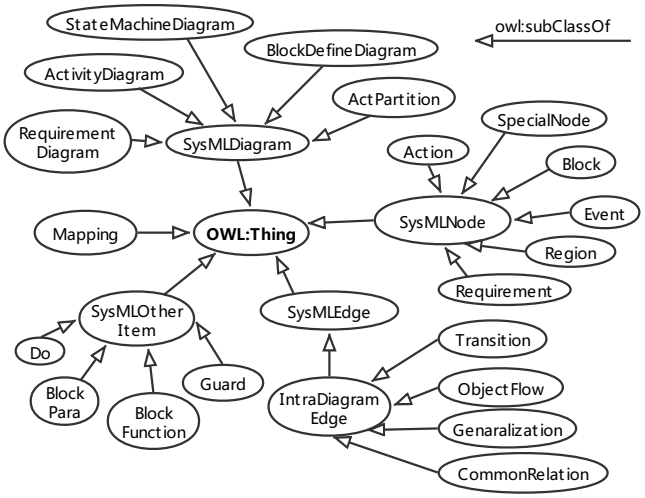


Fig. 1. Main parts of class hierarchy

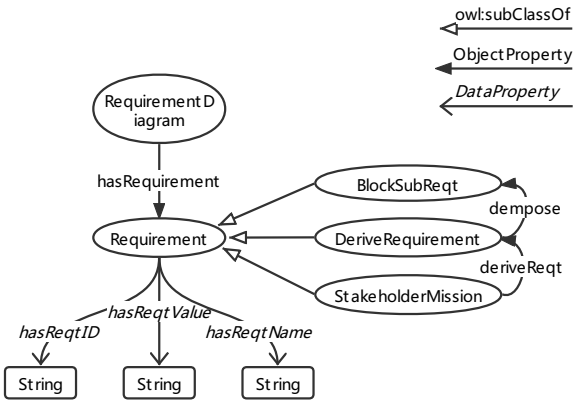


Fig. 2. Semantic representation of req

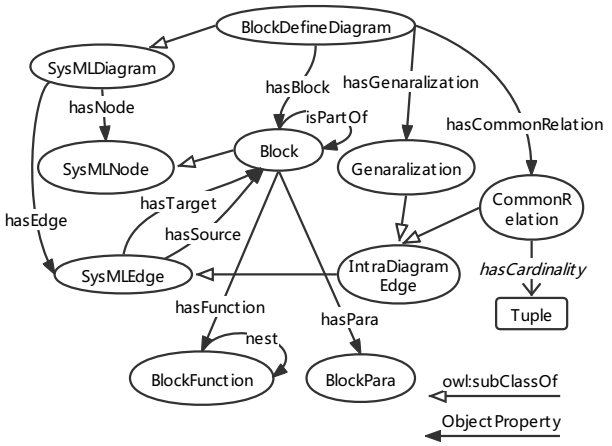


Fig. 3. Semantic representation of bdd

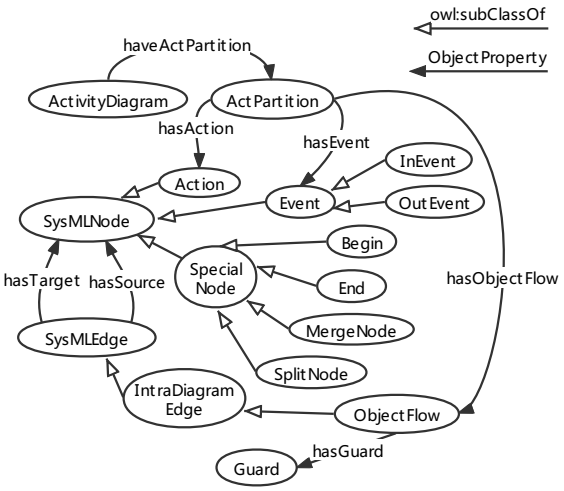


Fig. 4. Semantic representation of act

are usually visualized as a kind of diagram. Each diagram contains nodes and edges which represent various components of SysML and relationships between components. In addition to the vertical semantics of each diagram, there are the horizontal semantics between diagrams expressed through the cross-cutting mechanism. [7] extends cross-cutting mechanism with a data structure called Mapping for a more detailed traceability information model(TIM). To sum up, important terms include SysML diagram, node in diagram, edge in diagram, mapping, and other item of SysML component.

Step 3. Define the classes and their hierarchy This step first defines the most common concepts in the domain, and then specializes those concepts, i.e., a top-down development process [9]. As shown in Fig. 1, upper-level ontology $\bar{O}$ contains five main classes: `SysMLDiagram`, `SysMLEdge`, `SysMLNode`, `SysMLOtherItem`, and `Mapping` that implements traceability.

Step 4. Define the properties of classes. The classes alone can't provide enough information to answer the competency questions in Step 1. `ObjectProperty` and `DataProperty` are required to describe the internal structure of concepts. Fig. 2, 3, 4, 5 show the classes and properties that enable the semantics of requirement diagram(req), block definition diagram(bdd), activity diagram(act), and state machine diagram(stm), respectively.

We use `ObjectProperty` to represent edges with simple semantics e.g., `isPartOf` indicates edges in bdd that represent combination or aggregation. If using such a restriction on edges with complex semantics, such as transitions, we would need OWL Full. To keep the ontology as an OWL 2 DL ontology, transition is defined as a subclass of SysMLEdge, as shown in Fig. 1. Then transitions' extra semantics are represented by subclasses of `SysMLOtherItem`, then use the corresponding properties to connect them as shown in Fig. 5.

We need the second phase to complete the transformation from m to o.

Step 5. Enrich the class and class hierarchy. In a system, each block will have several particular instances; consequently, all blocks are modeled as children of `Block`.

Step 6. Create individuals. The final step is to create

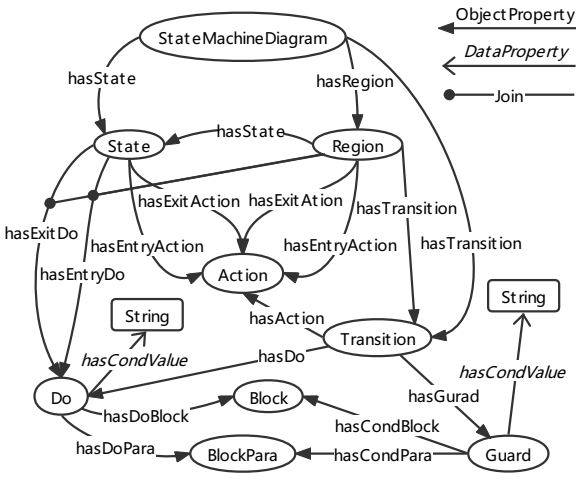


Fig. 5. Semantic representation of stm

individuals for the remaining instances in m. Creating an individual of a class requires (1) selecting a class, (2) creating an individual of that class, and (3) adding property values to that individual.

B. Traceability Information Model

The model in Fig. 6 specifies the well-formedness criteria for the potential traceability links between requirements and components. In practice, the ontology-based implementation of TIM provides at least two benefits [10]:

1) As tracing is a complex task, but the approach provides a guideline that simplifies its building and allows for flexible changes.
2) As traceability is implemented in a way that is friendly to people who did not create it and who only need to know some of the semantics of `ObjectProperty`.

The class hierarchy in *Requirement Concepts* demonstrates a process of modeling the analysis of requirements for safety-critical systems. `trace` establishes the traceability link between `Mapping` and `Block`, which is also the link point between *Requirement Concepts* and *Design Concepts*. If there exists a block b that satisfies the following axioms, then b is said to be a system component that satisfies the functional requirement q.

$$Block(?b) \wedge Mapping(?map) \wedge DeriveRequirement(?q) \\ \wedge hasMapping(?q, ?map) \wedge trace(?map, ?b)$$

Of course, it is possible that several b meet q. map have `mapTo` series of object properties to concretize the satisfaction. The details include that: (1)a block satisfies a requirement by a block function concretized by `mapToBF`; (2)a block satisfies a requirement by modifying its parameter concretized by `mapToBP`; (3)a block satisfies a requirement before or after an activity is performed in its activity partition concretized by `mapToAE`. `allocate` links a block to its activity partition, while `mapToBF` links a block function and an activity. These

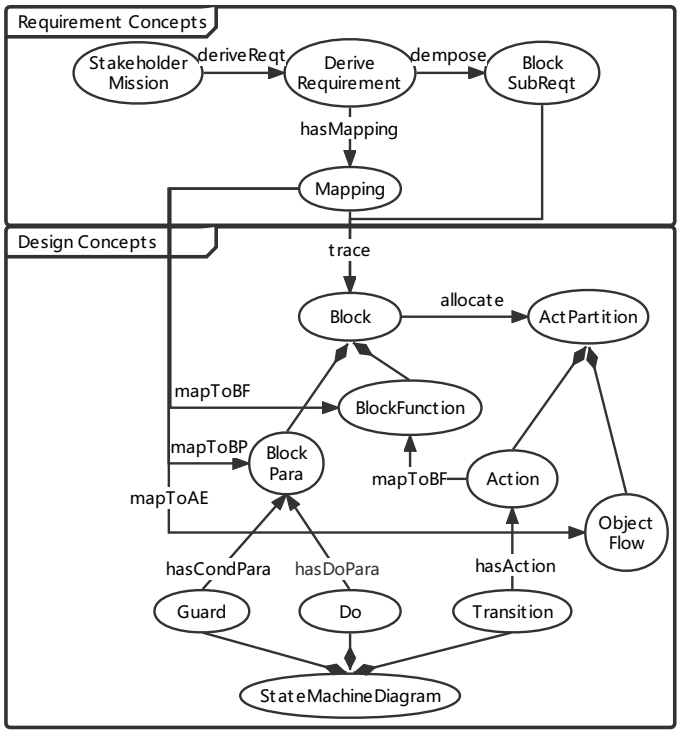


Fig. 6. Ontology_based TIM

mean that the function of a block can be performed as an activity in its activity partition. Conversely, functions that the block does not have cannot be performed. The final `has` series of object properties pass the link to the design of state machine diagrams.

III. Evaluation

A. Experiment Design

We first select a safety-critical system, Production Cell System (PCS) [11], and use the approach described above to obtain its lower-level ontology o_{PCS} as the experimental object. Production Cell System (PCS) is a well-known paradigm for embedded systems and was previously used as a baseline to evaluate the capabilities of various specification methods for safety analysis and verification. From the complete SysML requirements and design specification given in [11], it is known that PCS consists of 23 safety-related requirements and 6 main blocks, each with corresponding activity partitions and state machine diagrams.

We chose a comprehensive quantitative ontology evaluation method in [12], which has a study of metrics implemented in the popular quality frameworks. There are 8 sub-characteristics $RROnto$, $ANOnto$, $LCOMOnto$, $INROnto$, $CROnto$, $NOMOnto$, and $CBOnto$ which count the proportion of classes, properties, and individuals from different perspectives to measure the rationality of ontology design. Each sub-characteristic has a calculated value and a score out of five, based on each value. Based on the eight sub-characteristics, four more metrics are summarized to measure overall quality:

TABLE I
QUANTITATIVE EVALUATION RESULTS

Sub-characteristics		
Name	Value	Score
$RROnto$	0.64	4
$ANOnto$	1.00	5
$LCOMOnto$	2.44	4
$INROnto$	1.80	5
$CROnto$	14.80	5
$NOMOnto$	0.31	5
$CBOnto$	1.13	5
Metrics		
Name	Avg(scores)	
SE_v	4.33	
FAE_v	4.67	
ME_v	4.67	
Glo_v	4.50	

TABLE II
QUALITATIVE EVALUATION RESULTS

Dimension	Desription	Importance
Completeness	Missing domain or range in properties	Important
	Creating unconnected ontology elements	Minor
	Inverse relationships not explicitly declared	Minor
Compliance	No license declared	Important
Accuracy	Using a Miscellaneous Class	Minor

structural metric SE_v, functional adequacy metric FAE_v, maintainability metric ME_v, and global metric Glo_v. For details, see the framework presented in [12]. The qualitative evaluation tool we chose in [13] extends previous work on modeling errors and 41 pitfalls are identified with importance levels(critical, important, or minor). Experimental documents can be found at https://github.com/ch-wq81404/Experimental-documents.

B. Evaluation Results

Results of quantitative and qualitative evaluation are shown in Tab. I and Tab. II respectively. Tab. II shows the pitfalls and their importance in o_{PCS}. This will allow the user to correct the ontology and transform it into a better ontology.

From the results, almost all sub-characteristics get the highest score of 5, except for $RROnto$ and $LCOMOnto$. Anyway, the high scores of other metrics indicate the following:

- Rationalization of class richness portrayed by $CROnto$.
- Vertical and horizontal coordination of the class hierarchy portrayed by $LCOMOnto$ and $CBOOnto$.
- Quantitatively sufficient properties portrayed by $NOMOnto$, $INROnto$, and $ANOnto$.

It can be seen from Tab. II that o_{PCS} does not have problems of critical importance, which will affect the usage of ontology. All of these demonstrate the feasibility and efficiency of our approach.

IV. RELATED WORK

Existing transformation approaches are mostly limited to a single view [2], [5], [6] without traceability. [2] proposed the idea of using ontology to verify the dynamic behavior for complex systems. The work of [3] is closest to ours, and its ontology is used to analyze system change propagation. But they does not share the file so that our approach cannot be compared to theirs. All of them lack an automated tool.

V. CONCLUSION

We have presented a novel approach to maintain precious traceability between requirements and model multi-views design based on ontology. And the experimental results can be used by other relevant researchers to compare different transformation approaches or for other purposes. In future work, we will implement an automated tool to derive ontology model from SysML model, and select a different type of system as a case to explore the universality of our transformation method. In addition, we can use semantic inference tools of ontology to perform verification.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development under Project 2019YFA0706404, the NSFC under Project 61972150, and the Fundamental Research Funds for Central Universities.

REFERENCES

[1] O. M. Group, "How should SysML be applied to a MBSE project? How is SysML commonly abused?," https://sysml.org/sysml-faq/sysml-applied-mbse.html.

[2] C. Ruirui, Y. Liu, and X. Ye, "Ontology based behavior verification for complex systems." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 51739, 2018.

[3] H. Wang, V. Thomson, and C. Tang, "Change propagation analysis for system modeling using semantic web technology," *Advanced Engineering Informatics(AEI)*, vol. 35, pp. 17–29, 2018.

[4] S. Jenkins and N. Rouquette, "Progress on integrating owl and sysml,", NASA, 2012.

[5] H. Wardhana, A. Ashari, and A. Sari, "Transformation of sysml requirement diagram into owl ontologies," *Int J Adv Comp Sci Appl*, pp. 11, 2020.

[6] H. Graves, "Integrating sysml and owl," *Proceedings of OWL: Experiences and Directions*, 2009.

[7] B. Lionel, F. Davide, N. Shiva, S. Mehrdad and Y. Tao, "Traceability and sysML design slices to support safety inspections: a controlled experiment," in *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 1, pp. 1–43, 2014.

[8] W. W. Consortium(W3C), "Owl 2 web ontology language: Direct semantics (second edition)," https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/, 2012.

[9] N. F. Noy, D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology,", 2001.

[10] P. Mader, O. Gotel, and I. Philippow, "Getting back to basics: Promoting the use of a traceability information model in practice," In *Workshop on Traceability in Emerging Forms of Software Engineering(ICSE)*, pp. 21–25, 2013.

[11] T. E. Klykken, "A case study using sysml for safety-critical systems," *Master's thesis*, 2009.

[12] G. R. Rold an-Molina, D. Ruano-Ord as, V. Basto-Fernandes, and J. R. M endez, "An ontology knowledge inspection methodology for quality assessment and continuous improvement," *Data Knowledge Engineering(DKE)*, vol. 133, pp. 101889, 2021.

[13] P. María, A. Gómez-Pérez, and M. C. Suárez-Figueroa, "Oops!(ontology pitfall scanner!): An online tool for ontology evaluation." *International Journal on Semantic Web and Information Systems(IJSWIS)*, vol. 10, no. 2, pp. 7–34, 2014.

Development of a Domain Specific Modeling Language for Educational Data Mining

Eronita M. L. Van Leijden and
Alexandre M. A. Maciel
University of Pernambuco - UPE
Recife, Brazil
emlvl, amam@ecomp.poli.br

Andrêza Leite de Alencar
Federal Rural University of Pernambuco
Recife, Brazil
andreza.leite@ufrpe.br

Abstract

In data mining solutions, the data selection phase plays an essential role in the success of decision-making. The tools that operate at this phase need to cater to each domain's technical and management challenges. Using a Domain-Specific M odeling L anguage (DSML), w e found an alternative to abstract data and simplify the selection for Educational Data Mining (EDM) process. This work presents a graphic DSML to represent the problem. We used a case study methodology and implemented a CASE tool for the language evaluation. We acquired evidence that the proposed language simplifies the data selection phase for EDM because it solves the technical and management challenges addressed to this domain.

1 Introduction

Nowadays, there has been a growth in the Educational Data Mining (EDM) field of r esearch. This area leads with the development of methods that help examine to collect data from educational platforms. This area's main objectives are to understand how students interact in their learning environments and what they learn. So it turns possible to propose decision-making actions for better educational results [1].

Among the phases of the EDM process, there is the data selection phase. In this phase, the information is identified among existing data sets and considered during the modeling process. This phase includes choosing which data to collect and ensuring that the data is coherent with the phenomenon to be analyzed [2].

The tools that operate in this phase generally need to meet the challenges of the technical aspects of data processing. These challenges are associated with a large amount of data processing from different sources and the significant data heterogeneity with structured, semi-structured, and unstructured data [3]. In addition, it is necessary to confront two management challenges: First, these solutions should enable the reuse of the models to understand already known educational phenomena, such as performance prediction; detection of behavioral patterns; evasion indicators; etc.; secondly, it is necessary to provide conditions for academic analysts, who are not experts in data processing and analysis, conduct an analytical process [2].

Model-Driven Development (MDD) is a development paradigm that uses models as the primary artifact of the development process. In MDD the implementation is (semi) automatically generated from the models [4]. In constructing an MDD tool for a specific domain, it is needed to define its modeling language initially. Domain-Specific Modeling Language (DSML) makes it possible to create rules with high-level graphic and/or textual definitions. When applied in an MDD tool, it acts as a spelling and grammar checker, with validation to avoid syntax errors or typos [5].

This work aims to develop of a Domain-Specific Modeling Language for Educational Data Mining, in which the solution considers the technical and managerial challenges of this domain. For this, it was modeled a language and developed a prototype of an experimental case tool. For validate these artefacts a case study was realized using different versions of Moodle databases to validate this work.

2 Background

This section presents essential concepts necessary for proposed solution understanding.

2.1 Domain Specific Modeling Languages (DSML)

As one of the elements used on the MDD, Domain-Specific Modeling Languages enables the creation of rules with a high-level graphic and/or textual definition to be converted into a low-level language [4]. The definition of a DSML involves at least three aspects: the domain concepts and rules (abstract syntax); the notation used to represent these concepts—let it be textual or graphical (concrete syntax); and the semantics of the language [4].

The abstract syntax of a DSML is particularized by a metamodel, which is itself a model and describes the concepts of the language, the relationships among them, and the structuring rules that constrain the model elements and their combinations in order to respect the domain rules. The concrete syntax provides a realization of its abstract syntax as a mapping between the metamodel concepts and their textual or graphical representation. The semantics of a DSML is normally given with natural language. However, although

DOI reference number:10-18293/SEKE2022-137

users can normally deduce the meaning of most terms of a DSML, a computer cannot act on such assumptions [6].

2.2 Design Theories for Visual Notation

Design theories for visual notation provide the scientific basis for evaluating and designing visual notations. According to Moody [7], two approaches stand out: descriptive theory and prescriptive theory. The descriptive theory is used only to understand how and why the visual notations communicate (visual grammar). The prescriptive theory consists of a definition of explicit principles that deal with the design of visual notation, which handles the transformation of an unconscious process into a self-conscious process (visual vocabulary) [7].

The anatomy of a good visual notation consists of adding the definition of graphic symbols (visual vocabulary) to the rules of composition (visual grammar). To this end, Moody [7] suggests that some principles for designing and evaluating graphic symbols. The main ones are Semantic Transparency, Visual Expressiveness, Semiotic Clarity, Perceptual Discrimination and Graphic Economy [8].

2.3 Related Works

Two works were selected for this article because they deal with approaches related to structuring the input data of the EDM process.

The work of Magalhães Júnior [9] is a proposition of a data model that brings together indicators applicable in various educational phenomena. The solution lists the attributes used in the different EDM works. Under the focus of the phenomenon "student dropout," a catalog was developed based on an Entity and Relationship Diagram(DER) that served as a data integration point. After this step, the author performed new queries in this intermediate base to generate the file, which is input in the EDM process. Its objective was to reduce the efforts to select attributes and subsequent preparation of the data for the EDM.

Manhães [10] developed an architecture based on three layers according to EDM concepts: data layer, application layer, and presentation layer. Although cited, work does not explain how data collection was performed. The proposed solution describes an architecture layer destined for the "selected data", called Knowledge Repository. It stores the different student data models used as input on the EDM process, i.e., a cataloging of data sets able to be mined.

3 Proposed Language

The proposed modeling language represents the flow design to carry out the first phase of an EDM process - the data selection. It means the match between a field in a source database and a field of a target database passed through a visual notation. Next sections describe this language.

3.1 Language Rules

The DSML requirements necessary for the development of this research are presented below:

RQ1: To enable the use of data from different data sources of the educational platforms.

RQ2: To allow different composing and storing data: relational database, spreadsheets, data warehouse, log file, data stream, and web data, among others.

RQ3: To allow the cataloging of data structures by educational phenomenon (knowledge record).

RQ4: To allow the standardization of the "selected data" for the EDM process entry independent of the educational platform. It should also enable the data to continue as a file, a relational, or a multidimensional model.

3.2 Abstract syntax

The abstract syntax proposed in this work was inspired by Alencar [11]. It has similarities with the requirements RQ1, RQ2, and RQ4 listed in Section 3.1 and because of this, it was aggregated to the metamodel. The proposed metamodel is detailed in dissertation by Leijden [12] where the adjustments made are presented and the origin of new objects is explained.

3.3 Concrete syntax

The visual notation to represent the educational data selection process was developed following the main prescriptive principles proposed by Moody [7]. Here are the details of each of these principles and how they were considered in this work:

- **Semantic Transparency** – defines the visual representations used in a way that their looks suggest meaning. Area 2 of Figure 1 shows the list of defined graphic symbols and the labels for each proposed symbol.

  1. "Base Tool": represents the input data sources. It is a classic symbol for databases representation.

  2. "Base Version Tool": represents the input data sources. The "V" mark suggests it is a version of the database.

  3. "Entity Tool": is a regular blue pentagon with the letter "E" to represent the variants of the input information set. The letter "E" in the image represents an allusion to the entity in the ER model.

  4. "Mining Phenomenon": the mining cart with precious stones refers to an EDM process. The image depicts the data sources selected to perform the data mining process, polished to generate information.

  5. "Educational Phenomenon": the blue owl on a book represents the variants of the data sets selected to be mined. As the owl is a classic symbol representing

education (area of this research), the image represents the educational phenomenon as an entity.

6. ”Attribute Tool”: a red diamond with the letter “A” represents the characteristics of each entity. The letter A refers to the term attribute usually used in the ER model.

7. ”Sub Attribute Tool”: it is similar to Attribute Tool but uses different color (yellow). In addition, the ”greater than” sign was inserted to represent dependence with a specific Attribute Tool.

8. ”Association”: the blue double arrow represents relations between entities. The symbol was inspired by Bachman’s notation [13] which became known as arrow notation.

9. ”Flow”: arrows dashed in black. It represents the equivalence between the attributes of a source and those of a target entity. It is the component that shows the data flow.

- **Visual Expressiveness**-defines how should uses variables and visual capabilities and how to group strongly related elements. Area 2 of Figure 1 shows the elements distributed in 4 groups: ”Source Area”, ”Target Area”, ”Data Composition” and ”Event Composition”.

  ”Source Area”: group the elements that represent the ”entry area” concept.

  ”Target Area”: group the elements that represent the ”data to be mined area” concept.

  ”Data Composition”: group the elements related to characteristics and the relation of each entity of databases (entry and to be mined).

  ”Event Composition”: : the event symbols. The dashed arrow represents the flow that the mapped data goes between origin and educational phenomenon.

- **Semiotic clarity** - defines that there must be a 1:1 (one-to-one) correspondence between the semantic constructors (metamodel) and the graphic symbols of the language. Five metaclasses of the metamodel go mapping between the semantic constructors and the visual syntax: ”Base”, ”BaseVersion”, ”Attribute”, ”Association” and ”Mining Phenomenon”.

- **Perceptual Discriminability** - defines that different symbols must be clearly distinguished from each other. This principle is applied to the graphic symbols and the model diagramming when the graphical elements are inserted in the drawing area. It uses the container-based visual technique to demonstrate the hierarchy among the elements.

- **Graphic Economy** - defines that the number of different graphic symbols must be cognitively manageable. In this work, the ”Value” element does not have a symbol associated with its visual representation to limit the graphic and diagrammatic complexity of the model.

3.4 CASE Tool

Developed using Sirius , the CASE tool is organized into three regions (Figure 3). Region 1 consists of a drag-and-drop area for the compositions of a data selection case, Region 2 shows the components (symbols of the metamodel elements) placed in a palette tab, and Region 3 offers a properties tab for the selected objects in the drawing.

Performing data selection in a CASE tool based on metamodel and modeling language makes the tool automatically verify possible flaws in creating its routines regarding the use of symbols (visual vocabulary) and the composition rules (grammar). The CASE modeling tool can validate the diagrams, verifying that they follow the established syntax and semantics. This feature prevents users from misusing the model’s graphic symbols.

4 Analysis and Discussion

The analysis was conducted by following the methodological procedures for the study case presented by Yin [14]. It aimed to evaluate the modeling language, through a prototyped CASE tool, regarding the adequacy of its use in the circumstances of EDM projects, particularly in the first phase, which is the data selection. Table 1 shows the synthesis of the way the case study was conducted.

Table 1: Synthesis of the case study

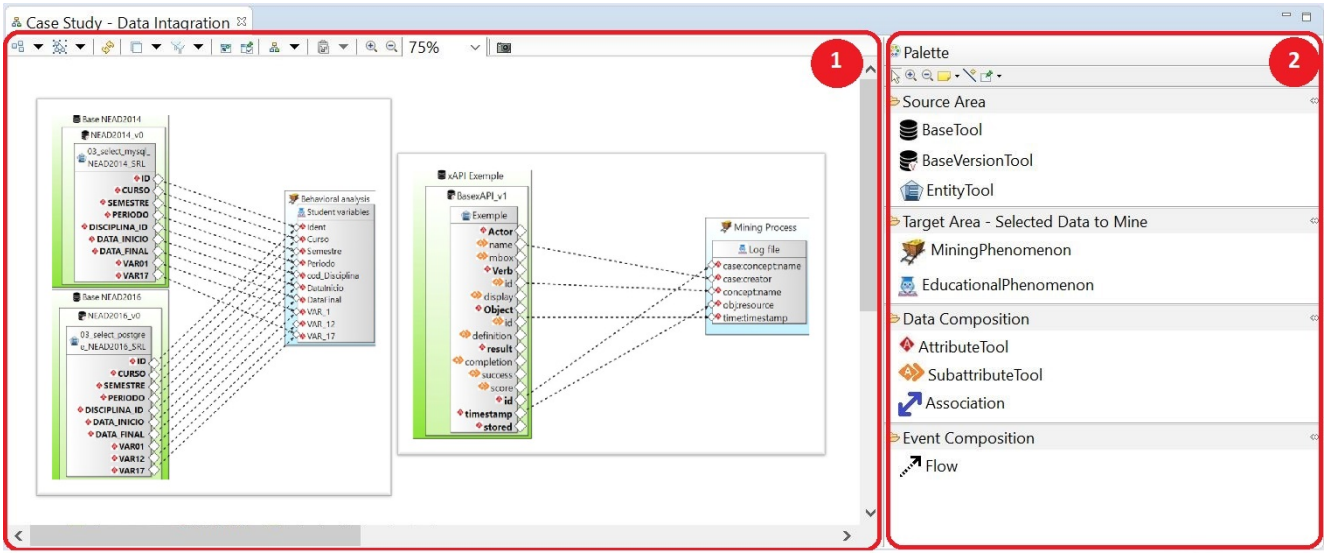
Description	Qualitative research of a descriptive character
Type Design	Single-Embedded
Study object	Language assessment using a prototype CASE tool
Study Unit	Problem situations in the data selection process
Data collect	Systematic observation
Data analysis	- Application of fragments of contentanalysis in two studies: consumption ofthe xAPI standard and unification of astructured database. - Fault simulations.

4.1 Context and Measured Variables

The observed variables are connected to the functional quality of the software. Following the model recommended by ISO/IEC 9126 (NBR13596) [15], evidence was observed regarding the variables of adequacy, accuracy, and interoperability.

The evidence about the adequacy and interoperability measures was obtained through observation when simulating the data selection process adopting the developed proto-

Figure 1: CASE Tool print screen



type and comparing the aspects observed with the requirements set out in Section 3.1. Figure 1, area 1, illustrates the diagrams created in this phase.

As for accuracy assessment, evidence was captured using the tool while executing modeling simulations that would violate the rules and restrictions created in the grammar developed in the modeling language.

4.2 Discussion

The planning and execution of the study case were carried out based on the objective of the work following research questions below:

Q1: Does the functional behavior conforms to proposed by the rule, the meta-modeling, and the language notation when using the CASE tool?

Q2: The requirements listed in section 3.1 met?

Q3: Can educational analysts, even not being data processing experts, increase the autonomy to carry out a data selection in the EDM process?

The results of these analyzes are depicted below.

Accuracy: During the modeling simulations, it was observed that the rules and restrictions placed on the visual modeling syntax (metamodel + language) were all taken into account. For instance, the tool, when correctly used, did not allow to create a flow from the input base to another input base; nor did it allow to create a data flow from the ”Mining Phenomenon” to the input base. Especially, the tool automatically made checks for flaws in the construction of the modeling that prevented the use of wrong model components in astray compositions in the process.

Adequacy: By applying the developed prototype, the result of the diagramming was evaluated, shown in Figure 3, with the requirements listed in Section 3.1. This analysis explicitly answers the research question Q2.

Requirement 1 and 2: It could be seen that the tool had achieved its goal since it was able to represent the modeling of both data from the xAPI standard, which is in JSON format, and of the Moodle database, which is a SQL structured query.

Requirement 3: For the two situations presented in the case study, the researcher chose to use a known data set structure, thus seeking to use the concept of knowledge reuse. Albeit it was possible to represent the modeling of the known data structure for analysis in EDM, this proposal does not guarantee such situation since the defined grammar only makes feasible a future development of an executable code that accesses some knowledge repository.

Requirement 4: Part of requirement 4, which deals with the issue of data representation in a semi-structured format, has not been directly validated.

Nonetheless, we can infer that this condition is valid by considering that the structure of the meta-modeling presented for the data referring to the source (which has been validated) is the same that will represent the selected data set, target base.

Interoperability: The tool can perform technical interoperability, as the solution covers two fundamental problems in information integration: data exchange and entity resolution [16]. In the tool, data exchange is promoted when the solution’s ability to represent different arrangements is demonstrated. As for entity resolution, the tool can identify and associate the information between data sources in a single destination, as depicted by the ”unification of structured databases” situation.

Given these analyzes, the specific research questions posed in Section 4.2 were considered and answered.

Regarding Q1, it was found to be true. Evidence was ac-

quired in the analysis of "accuracy" and, comprehensively, also obtained in the analysis of "adequacy". For the tool to perform the syntax's automatic validations, the metamodel must be defined according to the needs pointed out as requirements of section 3.1.

As for Q2, the answer is explicitly found in the "adequacy" analysis. The answer to Q3 is obtained while analyzing the creating process of each diagram. In neither of the two diagrams created was required the use of programming languages. Everything was done using clicks, moving graphic elements, and filling properties. This characteristic is inherent to the MDD technique. It demonstrates that non-expert users in data processing and analysis can conduct an analytical process (at least when it comes to the first phase of the EDM process).

5 Conclusions and Future Work

Throughout this work, it could be perceived, by empirical analysis, that the language created allows the diagramming of the phase of data selection to be used in EDM process, without the need for technological knowledge. In addition, the functional quality of the software was validated, as displayed in the observation on functional quality; adequacy, accuracy, and interoperability. Given what was brought and discussed, the work presented the following contributions:

Expressive metamodel - verification made when answering the Q1 of section 4.2, the functional behavior in the prototype was adherent to what was identified in the rules of language and what was proposed in meta-modeling and the language notation.

Cognitively effective notation - the work attends to the principles proposed by Moody [7].

Functionally adherent to the needs of the domain - demonstrated in detail in section 4.2, which deals with the analysis and discussion of the execution of the case study.

Simplification of the phase - verification made when answering Q3 in section 4.2.

As for future works, it is intended: i) to implement elements of transformation of the MDD. The T2M and M2T transformations in the following transformation functionalities: in the automatic data diagramming, where the data structure will be obtained from the source and automatically transformed into the model, and in the automatic generation of the source codes, so that the built model can be executed automatically at regular intervals by some task management tool like crontab, for example; ii) to expand the proposal to also carry out the pre-processing phase of EDM, promoting the solution to the ETL environment; iii) to complement the research validation, which may be an experiment, a participant observation and/or a questionnaire based on expert opinion; and iv) to develop studies of this proposal in the context of Big Data and Data Lake.

6 Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Mining in educational data: Review and future directions," in *Joint European-US Workshop on Applications of Invariance in Computer Vision*. Springer, 2020, pp. 92–102.

[2] C. Romero and S. Ventura, "Data mining in education," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 12–27, 2013.

[3] E. J. Dommett, "Understanding student use of twitter and online forums in higher education," *Education and Information Technologies*, vol. 24, no. 1, pp. 325–343, 2019.

[4] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis lectures on software engineering*, vol. 3, no. 1, pp. 1–207, 2017.

[5] J. White, J. H. Hill, J. Gray, S. Tambe, A. S. Gokhale, and D. C. Schmidt, "Improving domain-specific language reuse with software product line techniques," *IEEE software*, vol. 26, no. 4, pp. 47–53, 2009.

[6] H. Krahn, B. Rumpe, and S. Völkel, "Integrated definition of abstract and concrete syntax for textual languages," in *International Conference on Model Driven Engineering Languages and Systems*, vol. 4735, Springer. Berlin: Springer, 2007, pp. 286–300.

[7] D. L. Moody, "The" physics" of notations: a scientific approach to designing visual notations in software engineering," in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 2, IEEE. Cape Town: IEEE, 2010, pp. 485–486.

[8] H. B. M. Diniz, "Linguagem específica de domínio para abstração de solução de processamento de eventos complexos," Master's thesis, Universidade Federal de Pernambuco, 2016.

[9] P. N. Magalhães Júnior, "Um modelo de dados para apoiar a mineração de dados educacionais na investigação de evasão de estudantes," *tede.unifacs.br*, 2013.

[10] L. M. B. Manhães, "Predição do desempenho acadêmico de graduandos utilizando mineração de dados educacionais," *Doutorado em Engenharia de Sistemas e Computação Instituição de Ensino: Universidade Federal do Rio de Janeiro, Rio de Janeiro. Biblioteca Depositária: BIBLIOTECA DO CT*, vol. 1, no. 1, pp. 1–157, 2015.

[11] A. L. d. Alencar, "Um meta-modelo para representação de dados biológicos moleculares e suporte ao processo de anotação de variantes genéticas," *repositorio.ufpe.br*, vol. 1, no. 1, pp. 1–152, 2018. [Online]. Available: https://repositorio.ufpe.br/handle/123456789/32659

[12] E. M. L. V. Leijden, "Desenvolvimento de uma linguagem específica de domínio para consumo de dados educacionais," 2020.

[13] C. W. Bachman, "The structuring capabilities of the molecular data model." in *ER*, 1983, pp. 55–68.

[14] R. K. Yin, *Estudo de Caso-: Planejamento e métodos*. Bookman editora, 2015.

[15] I. ISO, "Iso standard 9126: Software engineering product quality, parts 1, 2 and 3," 2001.

[16] K. Qian, "Discovering information integration specifications from data examples," Ph.D. dissertation, UC Santa Cruz, 2017.

Agile and Lean Software Engineering and the SWEBOK

Position Paper for Panel: Educational and Professional Implications of SWEBOK

David Parsons
The Mind Lab
Auckland, New Zealand
david.parsons@themindlab.ac.nz

***Abstract*—This position paper addresses the usefulness (or otherwise) of the Software Engineering Body of knowledge (SWEBOK) version 3 for software practitioners in industry, and the consequent need for the SWEBOK to evolve to better address current industry practice. The position taken in this paper is that agile and lean methods are now the predominant approach to software engineering, and that the limited and anachronistic coverage of agile methods in the SWEBOK, coupled with the absence of any acknowledgement of lean approaches, is undermining software engineering education, the career prospects of graduates, and the software industry as a whole. It is therefore proposed that the agile methods section of the SWEBOK is revised and expanded such that it provides a valid body of knowledge for contemporary software engineering**

Keywords-Agile; Lean; Kanban; Scrum; DevOps; SWEBOK; SEEK

I. INTRODUCTION – SWEBOK, AGILE, AND LEAN

While the Software Engineering Body of Knowledge (SWEBOK) [1] has been guiding software engineering education for decades, it has continuously struggled to provide software engineering students with appropriate skills to excel in their jobs [2], and many graduates face difficulties when beginning their professional careers due to a skills mismatch between what is taught and what is needed [3]. One reason for this may be the failure of the SWEBOK to meaningfully provide a useful body of knowledge in contemporary practices in agile and lean software development. Of course, the SWEBOK is not the curriculum, but directly underpins it through the IEEE software engineering curricula guidelines [4]. Links between the Software Engineering Education Knowledge (SEEK) and the SWEBOK also strongly emphasize software engineering tools and methods [5], so both should reflect the dominant nature of agile methods in contemporary software engineering. This position paper first outlines the growth of agile methods over the last 20 years or so, along with the associated increase in the use of lean methods and practices such as Kanban, often integrated with agile processes. It then provides some commentary on the limitations of SWEBOK 3 in addressing these major trends and concludes by suggesting some ways in which the next version of the SEWBOK can improve its coverage in these areas.

A. The growth of agile methods

Agile methods, which emerged from lightweight methods in the 2000s, have continued to increase their influence over how software is engineered. Exactly how this increase has unfolded is to some extent unclear. One set of data suggests that uptake has been increasing markedly since about 2010 and is "now the norm" [6]. Snapshots using different data across different years provide varying perspectives. An internal Microsoft study in 2007 suggested that about a third of the teams were using agile methods [7]. Ten years later, a broader study of 153 practitioners gave a similar number, suggesting that about a third of organizations were using agile methods [8].

More recently, the 15th State of Agile Report [9] noted a significant growth in agile adoption, from 37% in 2020 to 86% in 2021. This seems in no small part to have been driven by DevOps initiatives, a complementary set of agile practices for iterative delivery in short cycles [10], which require the core agile practices of collaboration, automation, and tooling [11]. Of course, definitions and measures of agile adoption can be variable. A 2015 study by HP noted that only 15% of respondents claimed to be using "pure agile", while 51% were "leaning towards agile" [6]. A 2018 article in the Harvard Business Review noted that although about 40% of organizations had applied agile methods in parts of their operations, adoption was neither broad nor deep [12]. As Hoda et al. note, after more than two decades of agile practice, many organizations still consider themselves still maturing in this space [13].

These figures may suggest that, at the time of publication of SWEBOK 3 in 2014, agile methods had not yet reached the level of dominance in software engineering that they now appear to hold. In addition, the inconsistent application of agile methods suggests that better coverage in the SWEBOK might, through the improved knowledge of graduates and early career software engineers, lead to more mature usage in industry. In addition, software engineering education has already broadly embraced agile methods. A 2021 study found that 79.4% of software engineering education studies were associated with

DOI reference number: 10.18293/SEKE2022-060

Agile Software Development [2]. A revised SWEBOK would help to address this de facto move towards agile methods as a predominant software engineering approach.

B. Lean development

The application of lean thinking to software engineering is by no means as widespread or embedded as agile methods in industry. However, its links with agile methods, particularly in the sharing of practices such as Kanban boards in agile teams, and more explicitly in the Scrumban method, mean that we cannot fully address agile software engineering without at least acknowledging the influence of lean thinking. Its use is also growing significantly, for example the most recent "State of Agile" report shows that 22% of respondents were using some kind of lean approach. [9]. However, like agile methods, we must question to what extent this usage is broad and deep, with only a small number of organizations implementing Kanban beyond the "still maturing" stage [14]. Perhaps the SWEBOK can provide more support for this evolving area of software engineering?

C. Agile, Lean and SWEBOK 3

So, what of SWEBOK 3? There are a few references to agile scattered through the document, some of which raise questions about how it is categorized, for example '"Agile development" (actually an example of traditional incremental delivery)' (p. B-17) suggests a somewhat dismissive tone. The main agile methods section (one page out of 335) certainly shows its age, as we might expect from a document that is around ten years old. It refers to the most popular methods as being Rapid Application Development (RAD), eXtreme Programming (XP), Scrum, and Feature-Driven Development (FDD). It is doubtful that this was true even in 2014, and certainly is not the case now. The most recent state of agile report shows that 66% of respondents were using Scrum. Only 1% used XP, with no sign of RAD or FDD [9]. The agile development section of the SWEBOK continues with a discussion around combinations of agile and more plan-based methods, but this is neither referenced nor illustrated with any examples. Essentially the problem is that the SWEBOK does not provide any kind of body of knowledge for agile software development. Neither does it provide any support at all for an understanding of any aspects of lean development or DevOps.

II. A Proposal for SWEBOK 4 – Agile and Lean bodies of Knowledge

The practice of software engineering is evolving all the time. A systematic literature review by Garousi et al. (2019) revealed that professional practice and project management are becoming increasingly important and emphasize the soft skills that are essential to modern agile software development [3]. It is clear that both agile and lean software engineering are becoming increasingly popular, but also that usage is immature. The SWEBOK can contribute to addressing this problem by providing an improved body of knowledge that can help to ensure that the core principles of contemporary agile and lean development are properly understood by those entering the profession.

Of course there will always be calls to expand the SWEBOK, such as providing better coverage of testing, maintenance and configuration [15]. To some extent, new coverage can be introduced simply by recontextualizing what is already there, for example by creating a DevOps curriculum from various existing components of the SWEBOK [16]. However, the current single page entry for agile development is simply not sufficient for today's software engineering environment and must be revised and expanded.

References

[1] P. Bourque, R. E. Fairley, and IEEE Computer Society, *SWEBOK v.3.0: Guide to the software engineering body of knowledge*. IEEE, 2014.

[2] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, "Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends," *Journal of Systems and Software*, vol. 172, p. 110736, Feb. 2021, doi: 10.1016/j.jss.2020.110736.

[3] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, "Closing the Gap Between Software Engineering Education and Industrial Needs," *IEEE Software*, vol. 37, no. 2, pp. 68–77, Mar. 2020, doi: 10.1109/MS.2018.2880823.

[4] P. Bourque, R. E. Fairley, and IEEE Computer Society, *Guide to the software engineering body of knowledge*. 2014.

[5] S. Frezza, M. Tang, and B. J. Brinkman, "Creating an Accreditable Software Engineering Bachelor's Program," *IEEE Software*, vol. 23, no. 6, pp. 27–35, Nov. 2006, doi: 10.1109/MS.2006.156.

[6] J. Jeremiah, "Agile vs. waterfall: Survey shows agile is now the norm," *TechBeacon*. https://techbeacon.com/app-dev-testing/survey-agile-new-norm (accessed Feb. 20, 2022).

[7] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, Sep. 2007, pp. 255–264. doi: 10.1109/ESEM.2007.12.

[8] L. R. Vijayasarathy and C. W. Butler, "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?," *IEEE Software*, vol. 33, no. 5, pp. 86–94, Sep. 2016, doi: 10.1109/MS.2015.26.

[9] Digital ai, "15th Annual State Of Agile Report | Digital.ai," 2021. https://digital.ai/resource-center/analyst-reports/state-of-agile-report (accessed Feb. 20, 2022).

[10] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A Survey of DevOps Concepts and Challenges," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, Nov. 2020, doi: 10.1145/3359981.

[11] W. Luz, G. Pinto, and R. Bonifacio, "Adopting DevOps in the Real World: A Theory, a Model, and a Case Study," *Journal of Systems and Software*, vol. 157, 2019, doi: 10.1016/j.jss.2019.07.083.

[12] S. Panditi, "Survey Data Shows That Many Companies Are Still Not Truly Agile," *Harvard Business Review*, Mar. 22, 2018. Accessed: Feb. 20, 2022. [Online]. Available: https://hbr.org/sponsored/2018/03/survey-data-shows-that-many-companies-are-still-not-truly-agile

[13] R. Hoda, N. Salleh, and J. Grundy, "The Rise and Evolution of Agile Software Development," *IEEE Software*, vol. 35, no. 5, pp. 58–63, 2018, doi: 10.1109/MS.2018.290111318.

[14] N. Tsonev, "The First-Ever State of Kanban Report Is Live!," *Kanbanize Blog*, May 20, 2021. https://kanbanize.com/blog/state-of-kanban-report/ (accessed Feb. 20, 2022).

[15] V. Garousi, G. Giray, and E. Tuzun, "Understanding the Knowledge Gaps of Software Engineers: An Empirical Analysis Based on SWEBOK," *ACM Trans. Comput. Educ.*, vol. 20, no. 1, p. 3:1-3:33, Nov. 2019, doi: 10.1145/3360497.

[16] A. Capozucca, N. Guelfi, and B. Ries, "Design of a (yet another?) devops course," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, Mar. 2018, pp. 1–18.

Multi-Label Code Smell Detection with Hybrid Model based on Deep Learning

Yichen Li
School of Computer Science and Technology
Soochow University
Suzhou, China
Email: ycli1024@stu.suda.edu.cn

Xiaofang Zhang
School of Computer Science and Technology
Soochow University
Suzhou, China
Email: xfzhang@suda.edu.cn

***Abstract*—Code smell is an indicator of potential problems in a software design that have a negative impact on readability and maintainability. Hence, it is essential for developers to make out the code smell to get tips on code maintenance in time. Fortunately, many approaches like metric-based, heuristic-based, machine-learning based and deep-learning based have been proposed to detect code smells. However, existing methods, using the simple code representation to describe different code smells unilaterally, cannot efficiently extract enough rich information from source code. What is more, one code snippet often has several code smells at the same time and there is a lack of multi-label code smell detection based on deep learning. In this paper, we propose a hybrid model with multi-level code representation to further optimize the code smell detection. First, we parse the code into the abstract syntax tree(AST) with control and data flow edges and the graph convolution network is applied to get the prediction at the syntactic and semantic level. Then we use the bidirectional long-short term memory network with attention mechanism to analyze the code tokens at the token-level in the meanwhile. Finally we get the fusion prediction result of the models. Experimental results show that our model can perform outstanding not only in single code smell detection but also in multi-label code smell detection.**

***Index Terms*—Code smell, multi-label, code representation, hybrid model, deep learning**

I. Introduction

Code Smells indicate problems related to aspects of code quality such as understandability and modifiability, and imply the possibility of refactoring [1]. So Code smell analysis, which allows people to integrate both assessment and improvement into the software evolution process, is of great importance. Software engineering researchers have studied the concept in detail and explored various aspects associated with code smells, including causes, impacts, and detection methods [2].

Many approaches have been proposed to detect code smells. Traditionally, metric-based [3] and heuristic-based methods [4] use the manually designed regulations to extract the features inside the code. However, it's difficult for developers to reach an agreement on the appropriate rules and corresponding metrics. Machine-learning based methods [5], which apply Support Vector Machine, Naive Bayes and Logistic Regression, still have a long way to go to conquer problems of manually selected features and extra computation tools [6]. In recent years, a universally well-performing deep learning model [7] has been applied to code smell detection. In addition, the abstract syntax tree(AST) has been used to extract the syntactic features from the source code to detect the code smell [8].

Furthermore, multi-label code smell detection has attracted attention. Since the code snippet tends to have many code smells that may lead to potential problems, multi-label code smell detection means to find out all code smells inside the code snippet instead of one at a time. Guhhulothu et al. carried the experiment on a multi-label dataset of combining labels of two code smell datasets and Random-Forest was applied to detect two code smells at the same time [9]. All of the methods above solve the problem to some extent, but they all have the limitations below:

- The models just use code tokens or ASTs simply. Such methods will lose part of the information that helps recognize each code smell more efficiently.
- No one has proposed a model which can make the multi-label classification based on deep learning. Since the code snippet may has several code smells at the same time, it's necessary to propose an efficient and convenient model to find out code smells.

To address these limitations, in this paper we propose a **h**ybrid **m**odel with **m**ulti-**l**evel code representation(HMML). We first parse the AST from the source code and add the control and data flow edges [10] to get the code property graph. Then we apply the graph convolution network(GCN) [11] to learn information from the high dimensions at the syntactic and semantic level. Meanwhile, we use the bidirectional long-short term memory(LSTM) network with attention to analyze the code tokens at the token-level. Finally, we use the outputs of two models by weight to get the predication result. What is more, all of the models mentioned in this paper have been optimized to fit for the multi-label classification task. We apply our HMML method to 100 high-quality Java projects from Github. Better results have been achieved not only on multi-label code smell detection but on some single code smell detections.

The main contributions of this article are as follows:

- We propose a hybrid model that extracts the multi-level

DOI reference number: 10.18293/SEKE2022-077

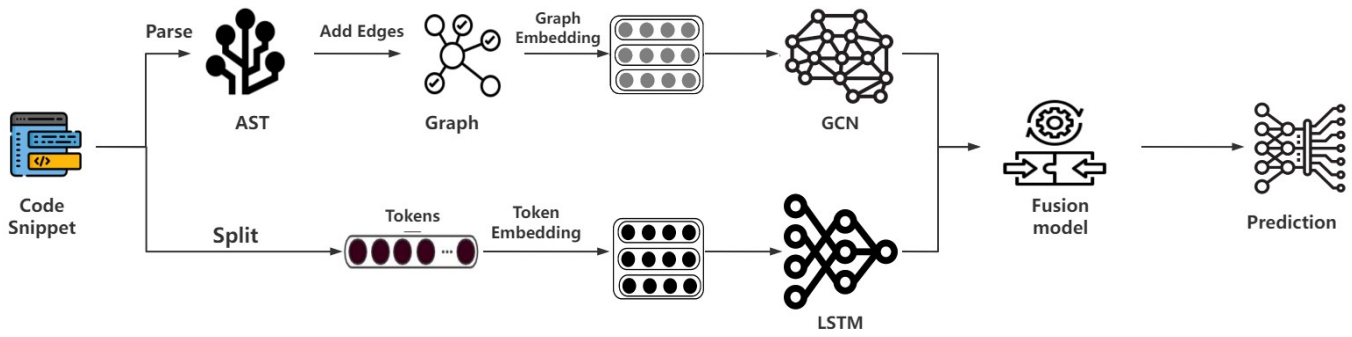


Fig. 1. Overview of the HMML

code representation information and separately applies the appropriate deep learning neural network.
- We are the first to carry out the multi-label code smell detection based on the deep learning method and achieve a good result.
- We modify many other approaches to fit into multi-label classification tasks and conduct extensive experiments to find the maximum capacity and best configuration.

The rest of this paper is organized as follows. Section II introduces the background; Our HMML method is introduced in Section III; Section IV describes the experimental setup and results are in Section V; The conclusion of this paper and the future work are presented in Section VI.

II. Background

A. Code smell

Code smells were first introduced by Fowler [1] as "structures with technical debt which affect maintainability negatively". Code smells imply the possibility of refactoring and have an impact on software development and evaluation. Fowler categorized code smells as implementation, design [12] and architecture [13] smells based on the scope and granularity [14].

B. Abstract syntax tree

Abstract Syntax Tree (AST) is a tree representation of the abstract syntactic structure of source code written in a programming language [15]. Developers can get the declaration statements, assignment statements, operation statements and realize operations by analyzing the tree structures [16].Nowadays, Some studies use AST-based approaches for source code clone detection [15], program translation [17], and code smell detection [8].

C. Motivation

Existing methods take a one-sided approach to the code smell detection problem. On the one hand, no one has applied the state-of-art deep learning to the multi-label code smell detection. On the other hand, many researchers focus on the token-based method [7] or AST-based method [8]. Although code fragments have some similarities with natural language texts and AST extracts some syntactic information, the information is still far from enough. Some code smells are caused by several aspects and the simple code representation fails to distinguish them. For example, *Long Method* is a general code smell and it is caused by the length of the code, long comment, complex conditional statement and messy loop. Existing methods cannot catch the cause of the code smell accurately because token-based methods ignore the syntax information by treating each code seperately and AST-based methods lose the words meaning and information about the comment, code length when compiling the code.

In the meanwhile, recent work has demonstrated the superiority of a graph-based approach to code representation over other approach [10]. Intuitively, the rich semantic and structural information in the graph will help us in smell detection. In terms of the *Missing default*, AST-based methods simply treat the statement as branch of the tree and ignore the possible logical errors linked to the data flow due to the missing default. By contrast, the graph-based methods with control and flow data can vividly show the change by adding extra edges among statements. To ensure the model's ability to catch different code smells, we fuse the token-based approach and graph-based approach to entirely get the structural, syntactic, semantic information to detect code smells.

III. APPROACH

This section introduces the method we use to detect code smells. Figure 1 gives an overview of our method.

To extract tokens and AST from Java programs, we use a python package javalang[1]. We use the method proposed in [10] to add the control and data flow edges. We focus on the following essential control flow types: *Sequential execution*, *Case statements*, *While* and *For loops*, which are linked to code smells mentioned in our motivation. In this paper, we use two different neural networks: a traditional LSTM for word tokens and a GCN that catches the information inside the graph.

1) LSTM Model: We use bidirectional long-short term memory network with attention mechanism to capture the

[1]https://github.com/c2snet/javalang

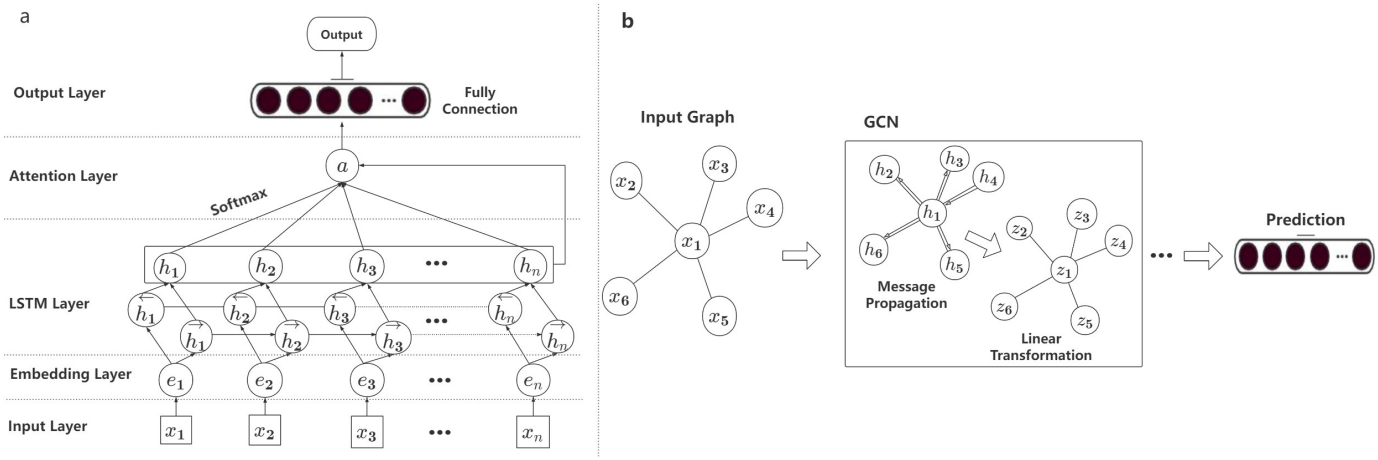


Fig. 2. Details of the Model

information in front and behind of the current position. Figure 2(a) shows details of the LSTM Model.

The attention is designed to selectively focus on parts of the source sentence during translation. We use global attention in this model to extract source context vector.

$$c_j = \sum_{i=1}^{|x|} a_{ij} h_i \tag{1}$$

where a_{ij} is the attention weights of hidden state h_i. The attention mechanism will give more weight to the hidden state vectors of important tokens.

$$r_{ij} = h_i * c_j \tag{2}$$

$$y = Sigmoid(W_s r_{ij} + b_s) \tag{3}$$

where W_s, b_s are parameters for Sigmoid layer. Here we use the Sigmoid layer as output layer to reveal the multi-label classification task.

2) GCN Model: Figure 2(b) shows details of the GCN Model. We use the python package PyG[2] to easily build a graph convolution network with the following propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{4}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections. I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and W^l is a layer-specific trainable weight matrix. $\sigma(.)$ denotes an activation function. $H^{(l)} \in R^{N \times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. Our forward model then takes the form below:

$$Z = \tilde{A}\ ReLU(\tilde{A}XW^{(0)})W^{(1)} \tag{5}$$

$$y = Sigmoid(W_s Z + b_s) \tag{6}$$

where W_s, b_s are parameters for Sigmoid layer,$W^{(0)} \in R^C$ is an input-to-hidden weight matrix for a hidden layer with H feature maps. $W^{(1)} \in R^R$ is a hidden-to-output weight matrix.

3) Fusion of Model: Assume the outputs of the model are o_1 and o_2 and the hyper parameter k , then the final probability distribution is computed as follows:

$$output = k \bigotimes o_1 + (1-k) \bigotimes o_2 \tag{7}$$

For the both models, we all use binary cross-entropy loss to optimize.

$$Loss(x_i, y_i) = -w_i[x_i log y_i + (1-x_i) log(1-y_i)] \tag{8}$$

where w_i is the parameter for loss, x_i is the i^{th} prediction of the label and y_i is the i^{th} ground truth.

IV. EXPERIMENTAL SETTINGS

A. Projects and dataset

We first use the CodeSplit[3] to split 100 high-quality Java projects on Github covering a variety of functions into method-level code fragments. Then we use Designite [18] to find out the smells contained in the source code and generate smell reports. Finally we choose nine code smells [18] at the method level for our experiment and combine their labels into a multi-label dataset. We divide all samples into three parts, 70% as the training set, 20% as the validation set, and 10% as the test set. Table I shows the number of samples used in our experiment and baselines.

B. Baseline

As mentioned before, we are the first to apply the deep learning methods to the multi-label code smell detection, and we select the following two improved methods as our baseline here, which are adapted into multi-label classification task:

1) Random–Forest Model: The model is used by [9] to reveal the multi-label classification and performs well when detecting *Long Method* and *Feature Envy*.

2) ASTNN Model: The ASTNN model is first introduced by Zhang [19] and was adapted by [8] to the single code smell detection. We refactor the model to do the multi-label code smell detection here.

[2]https://github.com/pyg-team

[3]https://github.com/tushartushar/CodeSplitJava

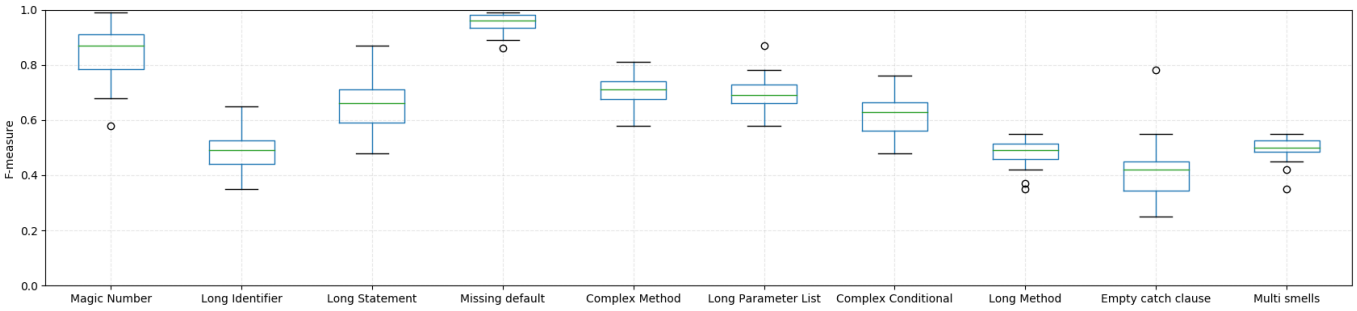


Fig. 3. Box plot of F-measure exhibit by HMML

TABLE I
SAMPLES DISTRIBUTIONS

	Training set		Validating set		Testing set	
Code smells	P	N	P	N	P	N
Magic Number	9643	76807	2748	21952	1387	10957
Long Identifier	926	85524	294	24406	132	12212
Long Statement	6816	69634	1955	22745	998	11346
Missing default	993	85457	308	24392	160	12184
Complex Method	2383	84067	713	23987	312	12032
Long Parameter List	1629	84821	33	24267	207	12137
Complex Conditional	1320	85130	346	24354	185	12159
Long Method	342	86108	92	24608	45	12299
Empty catch clause	558	85892	179	24521	75	12269
Multi smells	4972	81487	1475	23225	714	11630

C. Evaluation

Due to the extremely unbalanced distribution of positive and negative samples in real projects, we avoid comparing the accuracy of each model because if a model predicts all samples as negative, it will still have high accuracy. We choose precision, recall and F-measure as the evaluation metrics. For multi-label code smell detection, we use the Macro weighted F1 [20], which considers the imbalance in the category of samples. $Precision_{weighted}$ and $Recall_{weighted}$ are weighted according to the number of categories. Assuming L is the number of categories, they are defined as follows:

$$Precision_i = \frac{True\ Positive_i}{True\ Positive_i + False\ Positive_i} \tag{9}$$

$$Precision_{weighted} = \frac{\sum_{i=1}^{L} Precision_i \times w_i}{|L|} \tag{10}$$

$$Recall_i = \frac{True\ Positive_i}{True\ Positive_i + False\ Negative_i} \tag{11}$$

$$Recall_{weighted} = \frac{\sum_{i=1}^{L} Recall_i \times w_i}{|L|} \tag{12}$$

$$F1_i = \frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i} \tag{13}$$

$$Macro\ weighted\ F1 = \frac{2 * Precision_w * Recall_w}{Precision_w + Recall_w} \tag{14}$$

D. Training details

In our HMML method, the hidden states of LSTM have 300 dimensions and layer is set to be 2. We apply the graph convolution three times. In the two sub-models, the training batch size is set to be 32 and dropout is applied to avoid overfitting with dropout rate being 0.4. We use the Adam optimizer algorithm with 0.001 initial learning rate. In the ASTNN, we set two layers and 250 dimensions in the hidden states [19] Then we choose 80 features and 50 trees in the random forest [9]. Finally, we make our code public[4].

TABLE II
VALUES OF HYPER-PARAMETERS FOR HMML

Hyper-parameter	Values
Training batch size	{16,32,64,128}
Embedding dimensions(E)	{100,200,300}
Dimensions of hidden states in LSTM(H)	{150,250,300}
Number of layer in LSTM	{1,2,3}
Number of graph convolution in GCN	{1,2,3}

V. EXPERIMENTAL RESULTS

In this section, we mainly focus on answering the following research questions:

RQ1: How does our HMML method perform compared to other baselines?

RQ2: How does multi-label code smell detection perform compared with single code smell detection?

RQ3: What impact does each of our main components have in our HMML method?

A. RQ1:How does our HMML method perform compared to other baselines?

Table III shows the performance of our model and baselines on multi-label code smell detection and Figure 3 shows the box plot of performance of our HMML method under different configurations in Table II. From the table and figure, we can easily see that our model does not perform equally on all of the smells and it performs quite well on the smell like

[4]https://github.com/liyichen1234/HMML

TABLE III
PERFORMANCE OF HMML AND BASELINES ON MULTI-LABEL CODE SMELL DETECTION

	HMML			Random-Forest			ASTNN		
Code smells	P	R	F1	P	R	F1	P	R	F1
Magic Number	0.97	0.93	0.95	0.89	0.35	0.50	0.67	0.57	0.62
Long Identifier	0.44	0.55	0.49	0.52	0.36	0.43	0.85	0.30	0.44
Long Statement	0.73	0.60	0.66	0.90	0.35	0.50	0.84	0.68	0.75
Missing default	0.98	0.99	0.99	0.96	0.29	0.44	0.71	0.23	0.34
Complex Method	0.82	0.66	0.73	0.92	0.15	0.26	0.71	0.23	0.34
Long Parameter List	0.81	0.60	0.69	1.00	0.29	0.46	0.86	0.60	0.71
Complex Conditional	0.68	0.58	0.63	0.96	0.14	0.24	0.94	0.21	0.34
Long Method	0.67	0.41	0.51	1.00	0.09	0.16	0.83	0.69	0.76
Empty catch clause	0.51	0.30	0.38	0.86	0.08	0.15	0.61	0.11	0.18
Multi smells	0.83	0.74	0.78	0.90	0.30	0.45	0.76	0.52	0.62

Magic Number and *Missing default* and performs not bad on the other smells separately. In the meanwhile, the machine-learning method Random-forest has the poor Recall-values on each smell, which means the poor ability to find out the real code smell and the ASTNN performs unequally on the different smells.

In order to analyze the results, we apply the Win/Tie/Loss indicator to compare the performance of different models further, which has been used in prior works for performance comparison between different methods [8]. Then we conduct Wilcoxon signed-rank test and Cliff's delta test to analyze the performance of our model and other methods. Table IV shows Cliff's delta values($|\delta|$) and the corresponding effective levels.

To be specific, we make the following comparisons to determine the result of Win/Tie/Loss indicator: For a baseline method *M*, if our model outperforms *M* with the *p*-value of Wilcoxon signed-rank test less than 0.05 and the Cliff's delta value greater than or equal to 0.147, the difference between these two models is statistical significant and can not be ignored. At this time, we mark our model as a "Win." In contrast, if the model *M* outperforms out model with a *p*-value $<$0.05 and a Cliff's delta$\geq$0.147, our model will be marked as a "Loss." Otherwise, we mark the case as a "Tie".

TABLE IV
MAPPINGS BETWEEN CLIFF'S DELTA VALUES AND THEIR EFFECTIVE LEVELS

Cliff's delta	Effective levels
$\|\delta\| < 0.147$	Negligible
$0.147 \leq \|\delta\| < 0.33$	Small
$0.33 \leq \|\delta\| < 0.474$	Medium
$0.474 < \|\delta\|$	Large

As shown in the Table V, our method performs better almost in each smells and has an absolute advantage in multi code smells detection. Although weighted F1 can not accurately represent that the model can find all code smells at the same time, it reflects the ability of the model in multi-label code smell detection. Our HMML method has the highest weighted F1 and performs equally on the precision value and recall value. Therefore, we can regard that our HMML method does a good job in the multi-label code smell detection.

TABLE V
WIN/TIE/LOSS INDICATORS ON FMEASURE VALUES OF RANDOM FOREST, ASTNN, AND HMML

Code smell	Random Forest vs HMML	ASTNN vs HMML
Magic Number	$<$0.05(+Large)	$<$0.05(+Large)
Long Identifier	$<$0.05(+Medium)	$<$0.05(+Medium)
Long Statement	$<$0.05(+Large)	0.264(-Small)
Missing default	$<$0.05(+Large)	$<$0.05(+Large)
Complex Method	$<$0.05(+Large)	$<$0.05(+Large)
Long Parameter List	$<$0.05(+Large)	0.06(-Small)
Complex Conditional	$<$0.05(+Large)	$<$0.05(+Large)
Long Method	$<$0.05(+Large)	0.735(-Large)
Empty catch clause	$<$0.05(+Large)	$<$0.05(+Large)
Multi smells	$<$0.05(+Large)	$<$0.05(+Large)
Win/Tie/Loss	10/0/0	7/3/0

B. RQ2: How does multi-label code smell detection perform compared with single code smell detection?

TABLE VI
PERFORMANCE OF HMML AND BASELINES ON SINGLE CODE SMELL DETECTION

	HMML			Random-Forest			ASTNN		
Code smells	P	R	F1	P	R	F1	P	R	F1
Magic Number	0.98	0.94	0.96	0.88	0.36	0.51	0.85	0.85	0.85
Long Identifier	0.65	0.25	0.36	0.54	0.39	0.46	0.64	9.71	0.68
Long Statement	0.79	0.64	0.71	0.89	0.36	0.51	0.92	0.85	0.88
Missing default	0.88	0.84	0.86	0.93	0.34	0.50	0.83	0.72	0.77
Complex Method	0.84	0.83	0.84	0.84	0.15	0.26	0.85	0.25	0.39
Long Parameter List	0.86	0.72	0.79	1.00	0.50	0.66	0.86	0.58	0.70
Complex Conditional	0.79	0.50	0.61	0.93	0.14	0.24	0.83	0.75	0.79
Long Method	0.65	0.24	0.35	0.80	0.09	0.16	0.86	0.85	0.85
Empty catch clause	0.89	0.63	0.73	0.86	0.08	0.15	0.22	0.09	0.13

As shown in the Table VI, models show different abilities of detecting code smells. For each code smell, we apply the model used in multi-label code smell detection to train seperately and compare it with multi-label code smell detection model when detecting the designated code smell. Random-Forest performs equally in single code smell detection and multi-label code smell detection while ASTNN performs much better in single code smell detection. We believe this is somewhat related to capacity of ASTNN model, which cannot capture features of different code smells in the multi-label code smell detection. HMML performs slightly worse in multi-label code smell detection but still achieves a robust result.

What is more, single code smell detection needs to train corresponding model for each smell, which can be quite time consuming with the increase in the number of code smell. However, our multi-label code smell detection not only performs well in each code smell detection but can find out all code smells by one model at the same time.

C. RQ3: What impact does each of our main components have in our model?

We analyze the performance gain achieved due to various components of our approach by performing an ablation study.

Table VII shows these results. Control and data flow edges play a major role in the code smell detection. The reason is that code smells like *Empty catch clause* and *Missing default* which have the complex data flow information can be found out effectively in the GCN model. We can also find that the LSTM model and GCN model all perform bad on *Long Method*. This is because *Long Method* needs not only structural information but syntactic and semantic information.

TABLE VII
EFFECTIVENESS OF EACH MODULE IN HMML

	HMML^{-GCN}			HMML^{-LSTM}			HMML$^{-control\ and\ data\ flow\ edges}$		
Code smells	P	R	F1	P	R	F1	P	R	F1
Magic Number	0.98	0.89	0.93	0.87	0.85	0.86	0.83	0.81	0.82
Long Identifier	0.62	0.14	0.22	0.12	0.75	0.21	0.11	0.76	0.19
Long Statement	0.75	0.51	0.61	0.61	0.79	0.69	0.59	0.72	0.65
Missing default	0.89	0.79	0.84	0.99	0.96	0.97	0.77	0.74	0.73
Complex Method	0.90	0.59	0.71	0.75	0.66	0.70	0.70	0.63	0.66
Long Parameter List	0.80	0.41	0.54	0.85	0.71	0.77	0.86	0.73	0.79
Complex Conditional	0.77	0.58	0.66	0.77	0.58	0.66	0.75	0.42	0.54
Long Method	0.56	0.11	0.19	0.57	0.18	0.27	0.48	0.22	0.30
Empty catch clause	0.50	0.04	0.08	0.85	0.70	0.77	0.74	0.62	0.67
Multi smells	0.86	0.65	0.74	0.75	0.78	0.75	0.71	0.72	0.71

Fortunately, HMML notices the advantage and disadvantage of each model, which means the ability to catch appropriate features in multi-label code smell detection. HMML balances the results with the fusion of models and achieves a more robust result.

VI. THREATS TO VALIDITY

A. Internal validity

We use the Designite tool to detect smells, which is used to generate labels for the training data and view its results as ground truth. The tool uses three quotes to get more than 20 labels. Although the tool has been applied to many related works, it still needs much time to ensure the reliability of data.

B. External validity

We just did our detection on the 100 Java projects on Github. More jobs should be carried out on other projects, even transfer the model to the other languages since different languages may have its own distribution of code smells.

VII. CONCLUSION AND FEATURE WORK

In this paper, we propose a hybrid model, which extracts the multi-level code representation information to reveal multi-label code smell detection. Then we carry out the experiment based on the deep learning method and achieve a good result in terms of the evaluation.

As future work, a unified framework to deal with code smells at different granularities should be considered and we want to figure out whether existed approaches have the ability to find the unknown code smell. Moreover, it is of great value to make the model feasible to other programming languages.

VIII. ACKNOWLEDGEMENT

This work is partially supported by the National Natural Science Foundation of China (61772263, 61872177), Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

REFERENCES

[1] M. Fowler, *Refactoring - Improving the Design of Existing Code*, ser. Addison Wesley object technology series. Addison-Wesley, 1999. [Online]. Available: http://martinfowler.com/books/refactoring.html

[2] T. Sharma and D. Spinellis, "A survey on software smells," *Journal of Systems and Software*, vol. 138, pp. 158–173, 2018.

[3] M. Salehie, S. Li, and L. Tahvildari, "A metric-based heuristic framework to detect object-oriented design flaws," in *14th IEEE International Conference on Program Comprehension (ICPC'06)*. IEEE, 2006, pp. 159–168.

[4] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. Le Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 20–36, 2009.

[5] F. A. Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1143–1191, 2016.

[6] D. Di Nucci, F. Palomba, D. A. Tamburri, A. Serebrenik, and A. De Lucia, "Detecting code smells using machine learning techniques: are we there yet?" in *2018 ieee 25th international conference on software analysis, evolution and reengineering (saner)*. IEEE, 2018, pp. 612–621.

[7] H. Liu, J. Jin, Z. Xu, Y. Bu, Y. Zou, and L. Zhang, "Deep learning based code smell detection," *IEEE transactions on Software Engineering*, 2019.

[8] W. Xu and X. Zhang, "Multi-granularity code smell detection using deep learning method based on abstract syntax tree," in *Proceedings of the 33rd International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 07 2021, pp. 503–509.

[9] T. Guggulothu and S. A. Moiz, "Code smell detection using multi-label classification approach," *Software Quality Journal*, vol. 28, no. 3, pp. 1063–1086, 2020.

[10] W. Wang, G. Li, B. Ma, X. Xia, and Z. Jin, "Detecting code clones with graph neural network and flow-augmented abstract syntax tree," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 261–271.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[12] G. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for software design smells: managing technical debt*. Morgan Kaufmann, 2014.

[13] J. Garcia, D. Popescu, G. Edwards, and N. Medvidovic, "Identifying architectural bad smells," in *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE, 2009, pp. 255–258.

[14] T. Sharma, V. Efstathiou, P. Louridas, and D. Spinellis, "Code smell detection by deep direct-learning and transfer-learning," *Journal of Systems and Software*, vol. 176, p. 110936, 2021.

[15] C. Fang, Z. Liu, Y. Shi, J. Huang, and Q. Shi, "Functional code clone detection with syntax and semantics fusion learning," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 516–527.

[16] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "code2vec: Learning distributed representations of code," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, 2019.

[17] X. Chen, C. Liu, and D. Song, "Tree-to-tree neural networks for program translation," *arXiv preprint arXiv:1802.03691*, 2018.

[18] T. Sharma, P. Mishra, and R. Tiwari, "Designite: A software design quality assessment tool," in *Proceedings of the 1st International Workshop on Bringing Architectural Design Thinking into Developers' Daily Activities*, 2016, pp. 1–4.

[19] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 783–794.

[20] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

An Enhanced Data Augmentation Approach to Support Multi-Class Code Readability Classification

Qing Mi*, Yiqun Hao, Maran Wu, Liwei Ou
Faculty of Information Technology, Beijing University of Technology, Beijing, China
Email: miqing@bjut.edu.cn, yiqun.hao@ucdconnect.ie, 3085179961@qq.com, liwei.ou@ucdconnect.ie

***Abstract*—*Context:* Code readability plays a critical role in software maintenance and evolvement, where a metric for classifying code readability levels is both applicable and desired. However, most prior research has treated code readability classification as a binary classification task due to the lack of labeled data. *Objective:* To support the training of multi-class code readability classification models, we propose an enhanced data augmentation approach. *Method:* The approach includes the use of domain-specific data transformation and GAN-based data augmentation. By virtue of this augmentation approach, we could generate sufficient readability data and well train a multi-class code readability model. *Result:* A series of experiments are conducted to evaluate our augmentation approach. The experimental results show that a state-of-the-art multi-class code readability classification accuracy of 68.0% is reached with a significant improvement of 6.3% compared to only using the original data. *Conclusion:* As an innovative work of proposing multi-class code readability classification and an enhanced code readability data augmentation approach, our method is proved to be effective.**

***Index Terms*—code readability classification; data augmentation; generative adversarial networks; program comprehension; software analysis**

I. Introduction

Being a critical factor affecting the maintainability and reusability of the software, source code readability, defined as the ease of understanding the source code [5], is growing crucial in modern software development with a higher demand for rapid deliveries. Specifically, recent research reveals that software developers spend nearly 59% of their development time on reading and understanding the source code before they start coding [23]. Thus, it is worthwhile to provide a tool that constantly monitors the readability of source code and urges developers to write code with high readability to shorten the time wasted [21].

Buse and Weimer opened up the code readability classification research in early 2008 using machine learning algorithms with critical factors mainly affecting code readability [5]. Later, many effective classification models are built including the use of deep learning techniques [13][14]. However, code readability classification is still far from practical use. Because most prior researches treat code readability classification as binary classification [6][13][14][19], that is identifying a piece of code as either readable or unreadable. It is not precise enough and too extreme to be applied for practical use. Therefore, this paper is a pioneering work to propose a deep learning-based multi-class code readability classification model.

*Corresponding author.
DOI reference number: 10.18293/SEKE2022-130

However, the problem is that the size of the labeled readability dataset is merely acceptable for training a binary code readability classification model and far from being adequate for supporting a multi-class one. On top of that, using conventional methods to manually label data, such as conducting a large-scale survey to invite a number of programmers for data annotation, has always been too costly and inefficient [15]. Thus, we propose a data augmentation approach including a domain-specific data transformation method and a GAN-based (i.e. Generative Adversarial Network-based) data augmentation method to support the training of our deep learning-based multi-class code readability classification model.

The contributions of this paper are:

- To the best of our knowledge, we are the first to attempt multi-class code readability classification which classifies a code snippet as readable, neutral, and unreadable. This multi-class model is more suitable for practical use than existing ones.
- We propose a code readability data augmentation approach including domain-specific data transformation and GAN-based data augmentation. By utilizing the approach, labeled data could be generated from existing datasets along with a higher data diversity which could ultimately improve the performance of code readability classification models.
- We conduct a series of experiments to verify our proposed approach and gained a state-of-the-art multi-class code readability classification accuracy of 68.0%, f-measure of 67.3%.
- We publish all our data and source code online to benefit future researchers.[1]

This research is an extension of a short communication [15] which first proposed the use of data augmentation in code readability classification. There are several improvements made over the prior work:

- For the sake of enlarging the usable dataset to support multi-class classification, we refine the domain-specific data transformation method in a more systematic manner. Besides, WGAN rather than ACGAN is adopted in the GAN-based data augmentation method to get better stability.
- Apart from separately using two data augmentation methods, we propose a parallel augmentation method and a sequential augmentation method to combine the merits

[1] https://github.com/swy0601/Code-Augmentation

of them and further explore the best data augmentation method in the field of code readability classification.
- A series of experiments are conducted with more evaluation metrics and classification models to comprehensively evaluate our data augmentation approach and make the results more cogent.

II. Related Work

In general, past code readability classification researches fall into two categories: machine learning-based and deep learning-based. Although they both produce a classifier, deep learning-based models could automatically extract readability features whereas machine learning-based models rely on handcrafted features pre-specified by researchers.

A. Machine Learning-Based Code Readability Classification

Buse and Weimer collected 100 code snippets and invited 120 online human annotators to label them based on a five-point Likert scale ranging from one (i.e., very unreadable) to five (i.e., very readable). By analyzing the dataset collected, a set of handcrafted code features that correlates with code readability (e.g., average number of identifiers) was produced. Then, those features are fed into machine learning algorithms to make code readability predictions. This preliminary code readability model successfully outperformed human judgments on average [5].

Subsequently, Posnett et al. [18] proposed a readability classification model based on two factors: code size and entropy, which outperforms Buse's model on the same dataset. However, Dorn argues that both of those models have a bad generalization ability because they only take surface features into consideration [6]. Therefore, Dorn incorporated geometric, pattern-based, and linguistic features and built another machine learning-based readability model.

Scalabrino et al. [19] enrich Dorn's model metrics by complementing textual aspects and achieved a better performance than all prior models. Apart from a better model, Scalabrino also manually labeled and added 200 pieces of readability data which constitute our ground-truth code readability dataset along with data from Buse and Dorn.

B. Deep Learning-Based Code Readability Classification

While machine learning-based models have gained relatively high accuracy, they remain using handcrafted metrics which require a large amount of manual work. More critically, models with handcrafted features have a poor generalization ability to cope with more complex and realistic data. Concerning this issue, Mi et al. [14] put forward deep learning techniques which enable neural networks to directly learn features correlated to code readability from the source code and achieved a state-of-the-art classification accuracy of 82.8%.

Though binary classification reaches a high performance, the practicability is still not strong. Because it is too rough to judge a code snippet as either readable or unreadable in practice especially for some snippets that are just neutral in readability. Thus, we propose to explore multi-class code readability classification which has better practicability in realistic scenarios. On top of that, considering the effectiveness of deep learning techniques and the limited dataset, our research extends the use of data augmentation and deep learning techniques onto multi-class classification.

III. Proposed Approach

We treat code readability classification as a multi-class classification task with three categories: readable, unreadable, and neutral. As shown in Figure 1, our proposed approach consists of three main steps.

A. Dataset Construction and Code Representation

1) Dataset Construction: Collected by conducting a large-scale survey to invite annotators for labeling code snippets, open-source datasets from Buse [5], Dorn [6] and Scalabrino [19] are usually used as ground-truth data by most code readability studies. The final readability score is the average of every annotator's rank. Different from the previous binary classification [5][6][13][14][15][19], we partition dataset into three readability categories based on the readability score assigned (five-point Likert scale) to support multi-class classification. In addition, we remove code snippets in other languages and use only code snippets in JAVA which is consistent with prior researches [14][15]. In total, there are 420 labeled JAVA code snippets that make up our dataset. The top 25% code snippets with the highest readable score are considered as readable whereas the bottom 25% is considered as unreadable [15]. Therefore, the middle 50% is treated as the neutral readability data. We belive this partition conforms to reality because highly readable or unreadable code is less common than neutral ones. We finally split the gathered dataset into a training dataset and a test dataset in the ratio of 8:2.

2) Code Representation: A proper code representation method is important for deep learning-based code readability classification. Mi et al. proposed and deeply discussed three representation methods that could effectively capture code readability-related information [13]. A series of experiments were conducted to evaluate which code representation method is the most effective. Whereas results reveal that character-level representation has an outstanding capability surpassing the other two methods (with classification accuracy of 88.0%, 81.0%, and 75.5% respectively). Therefore, we adopt character-level representation in this paper. Specifically, code snippets are treated as two-dimension character matrices in which every letter, number, mark, and whitespace is converted into its corresponding ASCII value.

B. Code Augmentation

Considering conducting a large-scale survey to label new code snippets is too costly, we decide to apply advanced data augmentation techniques to enlarge our dataset.

1) Domain-Specific Data Transformation Method : The feasibility of using domain-specific data transformation in binary code readability classification was preliminarily disclosed in a previous research [15]. Thus, we propose to transform and

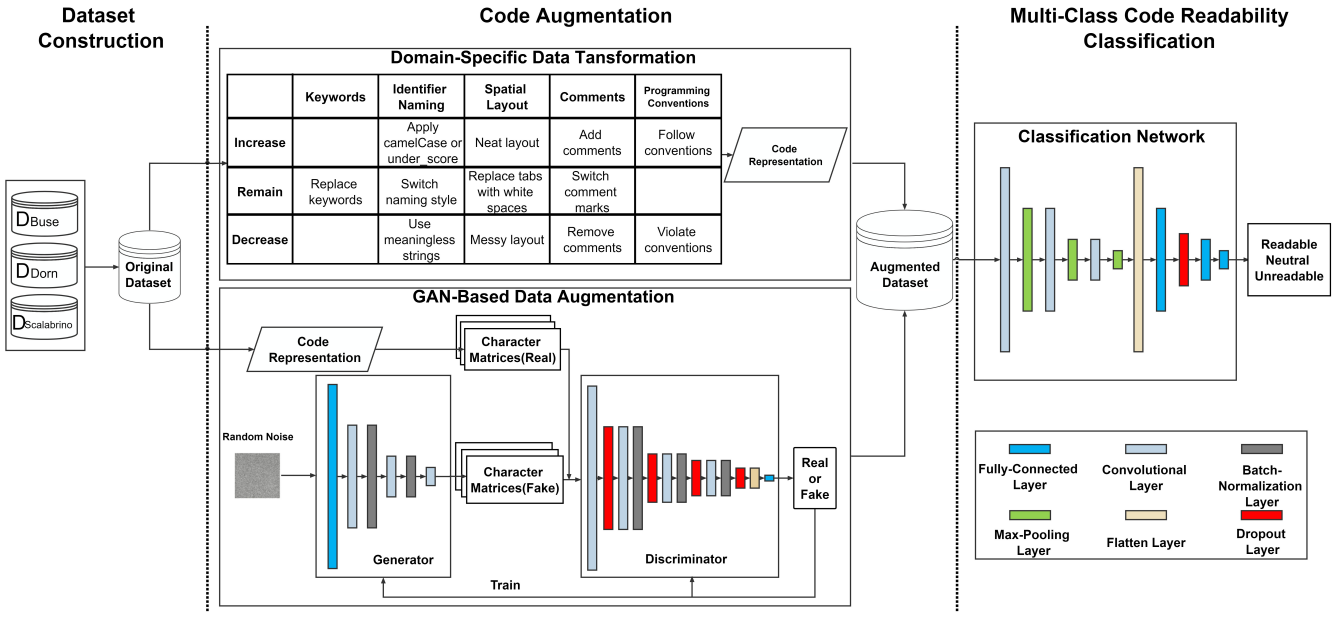


Fig. 1. Approach Overview

use it in multi-class classification. To adopt it in multi-class classification, we formulate three sets of operations as shown in Figure 1 that could generate new code snippets from the original ones without changing their label (which is the most secure way of augmentation because we cannot guarantee a correct output label after intentionally changing the original label). Specifically, we could perform increasing readability operations on readable data, decreasing readability operations on unreadable data, and remaining readability operations on neutral data to generate artificial data with the correct label.

2) GAN-Based Data Augmentation Method: Being able to generate artificial data out of a given dataset, GANs (i.e., Generative Adversarial Networks) have been proven to be a potent and efficient data augmentation method to compensate for the lack of data [3][7]. Specifically, we propose the use of the Wasserstein Generative Adversarial Network (i.e., WGAN) [2] for our task, because it could generate artificial data with a high diversity without the mode collapse and vanishing gradient problems [1].

Following the typical WGAN architecture, we construct our network as shown in Figure 1. The network is comprised of a generator and a discriminator. The generator could generate a character matrix from a given random noise, whereas the discriminator will determine if a given character matrix represents a real code snippet or a fake one generated by the generator. After adequate training, the generator should be able to generate verisimilar character matrices that could fool the discriminator and be treated as reliable artificially labeled data. To adopt it in our task, we put data with each readability label into training and generate new data with that label respectively. The detailed structures of the generator and the discriminator are introduced as follows.

- The generator starts with a fully-connected layer followed by two pairs of convolutional layers and batch-normalization layers. A convolutional layer is placed at the end. ReLU is set to be the activation function for all but not the last layer which uses Tanh as the activation. Furthermore, all convolutional layers use the same padding with the kernel size of 4 and all batch-normalization layers use the momentum of 0.8.
- The discriminator starts with a convolutional layer followed by three groups of dropout layers, convolutional layers, and batch-normalization layers. A dropout layer, a flatten layer, and a fully-connected layer are placed at the end of the discriminator. All convolutional layers use the same padding and the kernel size of 3 and all batch-normalization layers use the momentum of 0.8. The dropout ratio is set as 0.25. LeakyReLU with 0.2 as the alpha is used to be the activation function in this network.

During training the network, the loss function is Wasserstein Distance and the optimizer is RMSProp with the learning rate of 0.00005. We use the loss value to decide when to stop training. The output is scaled to integers in the range of -1 to 128 to get the same format as character matrices (see Section 3.1.2).

3) Parallel Augmentation Method: In this method, we use the two aforementioned methods, domain-specific data transformation and GAN-based data augmentation, to generate synthetic data separately, and then mix them to improve data diversity for training the classifier.

4) Sequential Augmentation Method: In this method, we first use domain-specific data transformation to generate synthetic data which is then used in the process of GAN-based data augmentation. After that, another batch of synthetic data is generated by GAN. Then, augmented data from both methods is mixed and used to train the classifier.

C. Multi-Class Classification Network

Considering the limited sample size, we propose a simple convolutional neural network [10]. The network starts with three pairs of convolutional layers and max-pooling layers. Then, there is a flatten layer followed by three fully-connected layers and a dropout layer as shown in Figure 1. RMS is used as the optimizer with a learning rate of 0.0015. Categorical cross-entropy is proposed to be the loss function [8].

IV. Experiment Setup

In this section, we present evaluation metrics and three research questions.

A. Evaluation Metrics

Considering that the number of code snippets varies in three readability levels, we propose to use the macro-accuracy and macro-f-measure, which are the most commonly used evaluation metrics in multi-class classification researches, to verify our experiment results. The evaluation metrics (Accuracy/Precision/Recall/F-measure) of different readability levels are directly added up for average, and all readability levels are given the same weight.

In addition, Brunner-Munzel test [4] is adopted as another evaluation metric to examine if there is a statistically significant difference between the results obtained with and without data augmentation. Furthermore, Cliff's δ effect size is used to quantify the magnitude of the measured difference.

B. Research Questions

Aiming to validate the effectiveness of our proposed approach, we formulate three research questions that will be answered through corresponding experiments. To improve generality, all experiments will be carried out in ten rounds.

RQ1: Which code augmentation method is the most effective for multi-class code readability classification?

Approach: We set the augmentation level[2] as N and 2N in this RQ because they are verified to be the most effective levels by the previous research [15]. Thus, we will compare the following four data augmentation methods with the augmentation levels of N and 2N:

- Domain-specific data transformation method
- GAN-based data augmentation method
- Parallel data augmentation method
- Sequential data augmentation method

RQ2: Which augmentation level is the best for multi-class code readability classification?

Approach: According to the result of RQ1, we would use the best data augmentation method to further probe the effect of different augmentation levels. Specifically, augmentation levels including 0N (no synthetic data), 1N, 2N, 3N, 4N, and 5N will be adopted and compared.

[2]Augmentation level stands for the ratio of augmented data to the original data, where N is defined as the total number of the original data

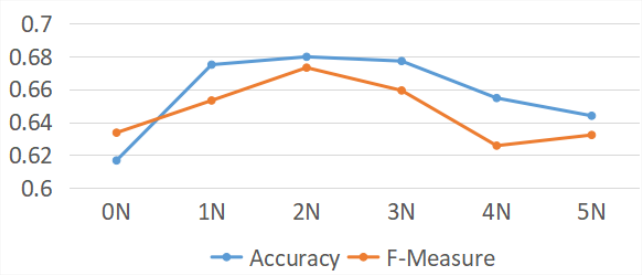


Fig. 2. Results of RQ2

RQ3: To what extent does data augmentation improve multi-class code readability classification?

Approach: Both the optimal augmentation method and level are used in this research question according to the results of RQ1 and RQ2. Instead of merely using the simple CNN (see Section 3.3) as the classification network, three other deep learning-based classifiers and two machine learning-based classifiers are chosen in order to explore the amount of improvement augmented data brought on different networks. In addition to accuracy and f-measure, the Brunner-Munzel test and Cliff's δ effect size [9] are also used in this RQ to provide a more intuitive and quantified evaluation revealing the improvement brought after the use of data augmentation.

V. Results

In this section, we present experimental results with respect to each RQ we proposed.

RQ1: Which code augmentation method is the most effective for multi-class code readability classification?

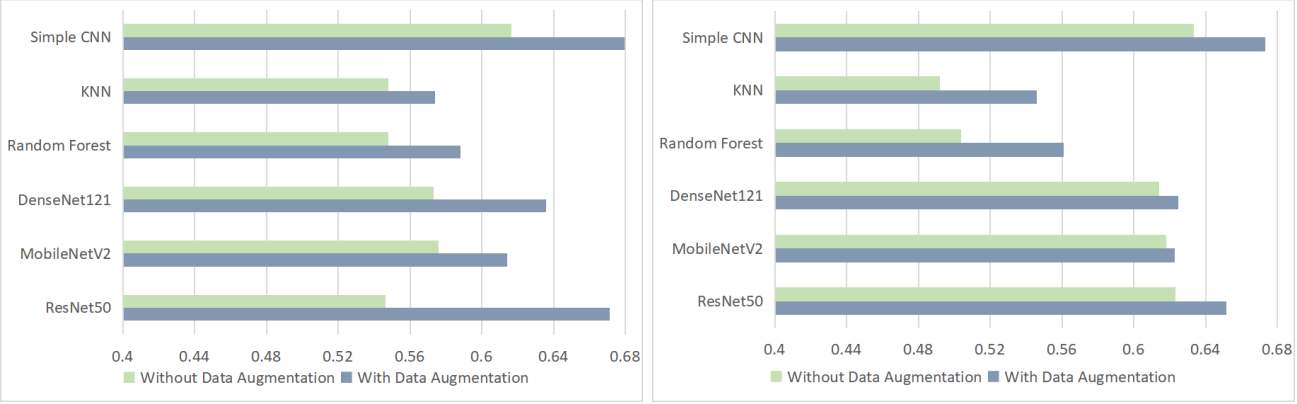
Based on the approach of RQ1, we repeat our experiments for ten rounds. The results are visualized in Table 1. It can be seen that the GAN-based data augmentation method outperforms the other three on both accuracy and f-measure. In contrast, the domain-specific data transformation method is comparatively inferior. We conjecture that snippets generated by the domain-specific data transformation method could not generalize beyond the original snippets which severely limits its performance. Whereas GAN could capture the readability features more accurately, thus improving the classifier performance. Besides, the two combined methods do not perform well. It might be due to data augmented by domain-specific transformation distracting the training of GAN which misinterprets the repeated parts as important and finally lower the quality of data generated by GAN. Considering that the GAN-based data augmentation method achieves a relatively better result in both evaluation metrics, we propose it to be the augmentation method used in RQ2 and RQ3.

RQ2: Which augmentation level is the best for multi-class code readability classification?

In this RQ, we evaluate the effect of different augmentation levels from 0N (i.e., only original data) to 5N. We conduct ten rounds of control experiments for each augmentation level.

TABLE I
RESULTS OF RQ1

Evaluation Metric	Without Augmentation	Domain-Specific Data Transformation	GAN-Based Data Augmentation	Parallel Data Augmentation	Sequential Data Augmentation
Accuracy	0.617	0.636 (1N) / 0.638 (2N)	0.675 (1N) / 0.680 (2N)	0.632 (1N) / 0.616 (2N)	0.623 (1N) / 0.658 (2N)
F-Measure	0.634	0.644 (1N) / 0.643 (2N)	0.653 (1N) / 0.673 (2N)	0.639 (1N) / 0.638 (2N)	0.625 (1N) / 0.659 (2N)



(a) Accuracy (b) F-Measure

Fig. 3. Results of RQ3

TABLE II
BRUNNER-MUNZEL TEST AND CLIFF'S δ EFFECT SIZE OBTAINED IN RQ3

Evaluation Metric	Simple CNN	KNN	Random Forest	DenseNet121	MobileNetV2	ResNet50
Brunner-Munzel test	0.00 (<0.05)	0.02 (<0.05)	0.00 (<0.05)	0.01 (<0.05)	0.15	0.00 (<0.05)
Cliff's δ effect size	0.68 (Large)	0.56 (Large)	0.84 (Large)	0.62 (Large)	0.38 (Medium)	0.83 (Large)

The results for accuracy and f-measure are shown in Figure 2. It can be observed that the optimal augmentation level is 2N in which the accuracy is 68.0% and the f-measure is 67.3%. Compared to the results where no augmented data is used, the improvements on accuracy and f-measure are significant with 6.3% and 3.9% respectively. However, a performance degradation appears when we increase the proportion of synthetic data over 2N. Thus, overusing artificially generated data is unhelpful and gives no further improvement. This conclusion falls in line with the prior paper [15], in which 1N and 2N are the best augmentation levels and there is also a decline in performance with too more synthetic data.

RQ3: To what extent does data augmentation improve multi-class code readability classification?

To comprehensively measure the improvement data augmentation brings, we select three other widely used deep learning-based classifiers, namely, RestNet50, DenseNet121, and MobileNetV2, and two machine learning-based classifiers, namely, random forest classifier and k-neighbors classifier. In terms of evaluation metrics, the Brunner-Munzel test and Cliff's δ effect size are also deployed to quantify the improvement along with accuracy and f-measure. Based on RQ1 and RQ2, we adopt the GAN-based data augmentation method with the augmentation level of 2N. Therefore, there are 1008 code snippets in the final training set and 84 code snippests in the final test set whereas the ratio of three readability labels remains 1:2:1 in both sets. The results are shown in Figure 3.

It is noticeable that data augmentation helps improve the performance of all classifiers. Because we did not fine-tune off-the-shelf deep learning-based models, they do not perform well than the simple CNN. The results of the p-value and the Brunner-Munzel test are shown in Table 2. It can be seen that all classifiers but MobileNetV2 have a p-value lower than 0.05 for the Brunner-Munzel test and a large d-value for Cliff's δ effect size. Both of them imply statistically significant improvements in classification accuracy.

VI. DISCUSSION

In this section, we discuss three worth noting aspects of our experiments.

Augmentation Effort. By adopting the data augmentation approach, the cost is largely reduced. The process of generating 5N (2100) artificial snippets by using the domain-specific data transformation was completed in two man-months. The process of generating data using GAN even saved more time and effort where it only took a day. Therefore, the data augmentation approach we proposed is both effective and efficient compared to the traditional way of conducting a large-scale survey to collect new data. However, such a survey is still necessary because all augmentation methods rely on the availability of ground-truth data.

Readability and Understandability. It is noticeable that the domain-specific data transformation does not perform well. We conjecture that it is due to the subjectivity code readability owns [5]. Because the entire transformation process is done by merely two people who might have a different

perspective on readability from the general people. Thus, data generated might have an inferior generality and limit the classifier's performance. Besides, manual manipulations might affect code understandability as well. Understandability is defined as to what extent the code allows developers to understand its purpose [11][20][22]. Readable code is not meant to be understood more easily. Thus, the domain-specific data transformation is only suitable for augmenting code readability data at the present stage.

Readability Data. The original dataset sourced from different prior researches [5][6][19] might be prone to errors and weaken our conclusions to a certain extent because three datasets are labeled by different groups of human annotators and differ remarkably in terms of code length, code completeness. For instance, the dataset collected by Scalabrino et al. is comprised of complete code snippets, whereas the other two datasets contain only partial snippets. Furthermore, the sample size might not be sufficient to train deep learning networks such as WGAN we used. As a result, GAN-based data augmentation could not be fully utilized and therefore produce ordinary performance. We believe that if the original dataset could be larger, the effectiveness of GAN-based data augmentation will be further improved. The shortage of data is also reflected in RQ3 where the simple CNN outperforms other complex networks due to over-fitting.

VII. Conclusions and Future Work

To enable multi-class code readability classification, we propose a data augmentation approach that could be used to effectively enlarge the dataset and well train a multi-class classifier. A series of experiments are conducted to evaluate the effectiveness of our proposed method. The results reveal that adopting the data augmentation approach could improve the classification performance to a considerable extent with the accuracy improved by 1.8% to 10.7%. The classifier produced by our proposed method could reach a state-of-the-art multi-class code readability classification accuracy of 68.0%, as well as 67.3% f-measure.

Our future work is mainly about improving the performance of the code readability classifier. We will try to utilize other code representation methods to capture more readability-related features such as code semantics. We will also improve the practicability of the code readability model. In fact, most text readability classification researches include more than three readability levels [12][16][17]. Therefore, we plan to increase the number of readability levels to be more realistic. Lastly, although considering data augmentation, the final size of the dataset is still too small for practical application because data augmentation highly relies on the original dataset. Therefore, labeling more data is still unavoidable to produce a high-performance code readability classifier.

Acknowledgments

This work was supported by the GHfund B (20220202, ghfund202202028015) and the Spark Project of Beijing University of Technology (Project No. XH-2021-02-24).

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[4] E. Brunner and U. Munzel. The nonparametric behrens-fisher problem: Asymptotic theory and a small-sample approximation. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 42(1):17–25, 2000.

[5] R. P. Buse and W. R. Weimer. Learning a metric for code readability. *IEEE Transactions on Software Engineering*, 36(4):546–558, 2009.

[6] J. Dorn. A general software readability model. *MCS Thesis available from (http://www. cs. virginia. edu/weimer/students/dorn-mcs-paper. pdf)*, 5:11–14, 2012.

[7] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[8] Y. Ho and S. Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2019.

[9] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong. Robust statistical methods for empirical software engineering. *Empirical Software Engineering*, 22(2):579–630, 2017.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] J.-C. Lin and K.-C. Wu. A model for measuring software understandability. In *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, pages 192–192. IEEE, 2006.

[12] M. Martinc, S. Pollak, and M. Robnik-Šikonja. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, 47(1):141–179, 2021.

[13] Q. Mi, J. Keung, Y. Xiao, S. Mensah, and Y. Gao. Improving code readability classification using convolutional neural networks. *Information and Software Technology*, 104:60–71, 2018.

[14] Q. Mi, J. Keung, Y. Xiao, S. Mensah, and X. Mei. An inception architecture-based model for improving code readability classification. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 139–144, 2018.

[15] Q. Mi, Y. Xiao, Z. Cai, and X. Jia. The effectiveness of data augmentation in code readability classification. *Information and Software Technology*, 129:106378, 2021.

[16] E. Miltsakaki and A. Troutt. Real time web text classification and analysis of reading difficulty. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 89–97, 2008.

[17] M. K. Paasche-Orlow, H. A. Taylor, and F. L. Brancati. Readability standards for informed-consent forms as compared with actual readability. *New England journal of medicine*, 348(8):721–726, 2003.

[18] D. Posnett, A. Hindle, and P. Devanbu. A simpler model of software readability. pages 73–82, 2011.

[19] S. Scalabrino, M. Linares-Vasquez, D. Poshyvanyk, and R. Oliveto. Improving code readability models with textual features. pages 1–10, 2016.

[20] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and R. Oliveto. Automatically assessing code understandability: How far are we? In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 417–427. IEEE, 2017.

[21] T. Sedano. Code readability testing, an empirical study. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 111–117. IEEE, 2016.

[22] M.-A. Storey, K. Wong, and H. A. Müller. How do program understanding tools affect how programmers understand programs? *Science of Computer Programming*, 36(2-3):183–207, 2000.

[23] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li. Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering*, 44(10):951–976, 2017.

Contrastive Learning for Multi-Modal Automatic Code Review

Bingting Wu
School of Computer Science and Technology
Soochow University
Suzhou, China
20204227028@stu.suda.edu.cn

Xiaofang Zhang
School of Computer Science and Technology
Soochow University
Suzhou, China
xfzhang@suda.edu.cn

***Abstract*—Automatic code review (ACR), aiming to relieve manual inspection costs, is an indispensable and essential task in software engineering. The existing works only use the source code fragments to predict the results, missing the exploitation of developer's comments. Thus, we present a Multi-Modal Apache Automatic Code Review dataset (MACR) for the Multi-Modal ACR task. The release of this dataset would push forward the research in this field. Based on it, we propose a Contrastive Learning based Multi-Modal Network (CLMN) to deal with the Multi-Modal ACR task. Concretely, our model consists of a code encoding module and a text encoding module. For each module, we use the dropout operation as minimal data augmentation. Then, the contrastive learning method is adopted to pre-train the module parameters. Finally, we combine the two encoders to fine-tune the CLMN to decide the results of Multi-Modal ACR. Experimental results on the MACR dataset illustrate that our proposed model outperforms the state-of-the-art methods.**

***Index Terms*—automatic code review, dataset, contrastive learning, multi modal, abstract syntax tree**

I. Introduction

Code review is a critical part of software maintenance and evaluation. A general process of code review is shown in Fig. 1. Whenever the developer has submitted the revision code and the comments, the system will schedule an appropriate reviewer who will need to compare the differences between the original file, the modified file and combine the content of the developer's comments to verify whether the code meets the requirements. However, it also has a great demand for human resources [1], which makes it impossible to expand code review on a large scale.

Therefore, many researchers are committed to Automatic Code Review (ACR) to solve the problem [2]–[4]. For ACR, they provide the model with the original code and the revised code, and the model returns the suggestions on whether this modification is acceptable.

In this paper, we present a new perspective that the ACR should not only contain code fragments. The developer's comments, explaining why and how to change code, are also an essential source of information in the code review process. The model should fully consider the developer's comments when giving suggestions for code review. Therefore, we focus on the Multi-Modal ACR task. However, there are three main challenges: 1) there is no corresponding dataset, 2) most of the existing models ignore the problem of few-shot, and 3) there is no multi-modal model in the Multi-Modal ACR task.

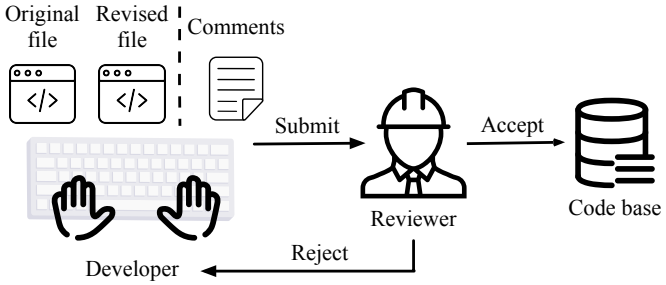


Fig. 1: Traditional code review process.

Since there is still a lack of relevant public datasets, we propose and open-source a Multi-Modal Apache Automatic Code Review dataset (MACR[1]), with 11 projects and over 500 thousand samples.

Recent works [2]–[5] have shown that deep learning methods perform better in capturing the syntactical and semantic information of the source code, enabling suitable code review suggestions. Among them, Shi et al. [2] proposes a method called DACE, which uses long short-term memory (LSTM) and convolutional neural network (CNN) to capture the semantic and syntactical information, respectively. Zhang et al. [5] proposes a method called ASTNN. The main idea is to obtain better feature extraction ability by dividing the abstract syntax tree (AST) into sentence-level blocks. Although these approaches are excellent in modeling code fragments, they do not consider the problem of few-shot. Misclassification may occur when the model encounters code fragments that have not been seen before. Therefore, in this paper, we adopt contrastive learning to obtain a uniform distributed vector representation to improve the robustness of the model.

In addition, we explore a new method CLMN, to represent code snippets and textual content. We first model the code snippets and textual content separately using two independent encoding modules SimAST-GCN [4] and RoBERTa [6], respectively. Since RoBERTa already has ideal pre-training

DOI reference number: 10.18293/SEKE2022-022

[1]https://github.com/SimAST-GCN/CLMN

parameters, we need to pre-train SimAST-GCN. We adopt the method of contrastive learning to train the parameters of the SimAST-GCN module. After finishing the pre-training of the two encoders, we concatenate the vector representations of two encoders to obtain a joint representation of code and text to predict the final code review result.

Experiments results show that the proposed methods achieve significantly better results than the state-of-the-art baseline methods on MACR datasets. Our main contributions are summarized as follows:

- We provide a large-scale Multi-Modal Apache Automatic Code Review dataset (MACR) for the Multi-Modal ACR task.
- We propose a novel neural network CLMN to learn the combined relationship of the code fragments and textual content to improve the accuracy of Multi-Modal ACR.
- We conduct a series of large-scale comprehensive experiments to confirm the effectiveness of our approach.

The rest of this paper is organized as follows. Section II introduces the background. Section III describes our approach. Section IV provides our experimental settings. Section V presents the experimental results and answers the research questions. Finally, Section VI concludes our work.

II. Background

A. Code Review

The general process of traditional code review is shown in Fig. 1. In traditional approaches, researchers cannot solve the main challenge in code review: understanding the code [7]. Therefore, researchers can only improve efficiency from other aspects of code review, such as recommending suitable reviewers [8], [9] and using static analysis tools [10], [11].

With the development of deep learning, we can understand the code by modeling the source code [2]–[4]. However, these methods only use code fragments to predict the results, lacking the exploitation of developer's comments in code review.

To solve the Multi-Modal ACR task, the model first extracts features from the original file, the revised file, and the comments. Then, the model needs to encode these features into vector representations and combine the code representation and the comment representation into a uniform representation. Finally, the neural network can make the final suggestion about the code review according to the uniform representation.

B. Abstract Syntax Tree

An abstract syntax tree (AST) is a kind of tree aimed at representing the abstract syntactic structure of source code. AST has been widely used as a vital feature in deep learning [2], [4], [12]. For example, it has a wide range of applications in source code representation [5], defect prediction [13], and other fields. Each node of an AST corresponds to constructs or symbols of the source code. On the one hand, compared with plain source code, ASTs are abstract and do not include all details such as punctuation and delimiters. On the other hand, ASTs can be used to describe the lexical information and the syntactic structure of the source code. In this paper, we use the Simplified AST [4] as the input for the code encoder module.

C. SimAST-GCN

In our previous study, we proposed a method SimAST-GCN [4]. SimAST-GCN first parses the code fragment into the AST and uses a specially designed algorithm to simplify the AST into a Simplified-AST. Then, we use preorder traversal algorithm to get the node sequence $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}]$ and the relation graph $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, where n is the number of nodes. Next, the representations for nodes are obtained by:

$$\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\} = \text{Bi-GRU}(\mathbf{x}) \tag{1}$$

Then, we use GCN to aggregate the information from their neighborhoods.

$$\mathbf{h}^l = \text{GCN}(\boldsymbol{A}, \mathbf{h}^{l-1}) \tag{2}$$

where $\mathbf{h}^l$ is the graph hidden status, $\mathbf{h}^0$ is the $\mathbf{H}$. Finally, we adopt a retrieval-based attention mechanism [14] to combine $\mathbf{H}$ and $\mathbf{h}^l$ to get the final representation of the code fragment.

D. Contrastive learning

Contrastive learning aims to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors. It assumes a set of paired samples $\mathcal{D} = \{(x_i, x_i^+)\}_{i=1}^m$, where x_i and x_i^+ are semantically related. We follow the contrastive framework in Chen et al. [15] and take a cross-entropy objective with in-batch negatives [16] : let $\mathbf{h}_i$ and $\mathbf{h}_i^+$ denote the representation of x_i and x_i^+, the training objective for (x_i, x_i^+) with a mini-batch of N pairs is:

$$\ell = -\log \frac{e^{sim(\mathbf{h}_i, \mathbf{h}_i^+)/\tau}}{\sum_{j=1}^N e^{sim(\mathbf{h}_i, \mathbf{h}_j^+)/\tau}} \tag{3}$$

where τ is a temperature hyperparameter and $sim(\mathbf{h}_1, \mathbf{h}_2)$ is the cosine similarity $\frac{\mathbf{h}_1^\top \mathbf{h}_2}{\|\mathbf{h}_1\| \cdot \|\mathbf{h}_2\|}$. In this work, we encode code fragments using a graph-based model SimAST-GCN and then train all the parameters using the contrastive learning objective (Eq. 3). For the text encoder, we use the pre-trained model RoBERTa and the parameters are trained by SimCSE [17].

III. Proposed Approach

We introduce our approach CLMN in this section. As shown in Fig. 2, the framework of the proposed CLMN contains two main components: the Code Encoder (SimAST-GCN) and the Text Encoder (RoBERTa). For each encoder, we use contrastive learning to learn an adequate representation of code fragments and the developer's comments. For the Code Encoder, we use an unsupervised contrastive learning method to train its parameters. During the training process, we use dropout as noise to obtain positive samples. The process is shown in Fig. 3. For the Text Encoder, we use the parameters trained by SimCSE directly. After the unsupervised training for the two encoders, we use the pre-trained Encoders to fine-tune the finally model CLMN to predict the results.

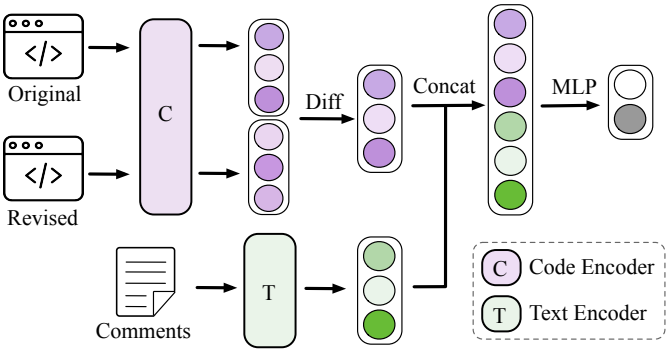


Fig. 2: General framework of CLMN.

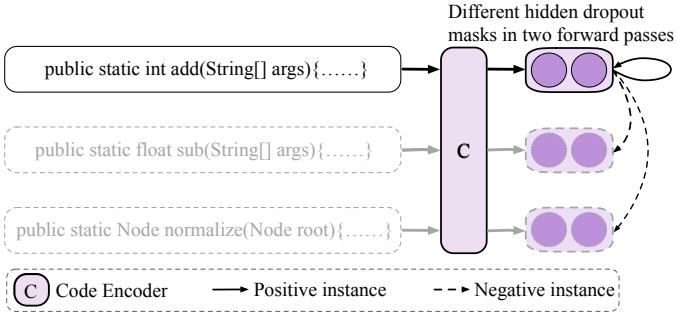


Fig. 3: Unsupervised Code Encoder predicts the input code fragment itself from in-batch negatives, with different hidden dropout masks applied.

A. Contrastive Learning for Code Encoder

The main challenge for contrastive learning is how to get positive samples. As shown in Fig. 3, in the Code Encoder, we pass the same code fragment to the model SimAST-GCN twice: by applying the standard dropout twice, we can obtain two different embeddings as "positive pairs". Then we take other code fragments in the same mini-batch as "negative" samples, and the model predicts the positive one among negative samples. Although it may appear strikingly simple, this approach is widely used in natural language models [17].

In this work, we take a collection of code fragments $\{x_i\}_{i=1}^{m}$ and use $x_i^+ = x_i$. We use the independent dropout for x_i and x_i^+ to make it work. In SimAST-GCN, there are dropout placed on each GCN layers(default $p = 0.1$). We denote $\mathbf{h}_i^z = f_\theta(x_i, z)$ where z is a random mask for dropout. We feed the same input to the encoder twice and get two embeddings with different dropout masks z, z', and the training objective of Code Encoder becomes:

$$\mathcal{L} = -\log \frac{e^{sim(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{sim(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'})/\tau}} \tag{4}$$

for a mini-batch of N code fragments. Note that z is just the standard dropout mask in SimAST-GCN and we do not add any additional dropout.

B. CLMN

1) Encoder Module: In our CLMN model, we have two pre-trained encoder modules, code encoder (SimAST-GCN) and text encoder (RoBERTa). For the code encoder, we denote $C = f(c)$ where c is a code fragment and C is the representation for this code fragment. For the text encoder, we denote $T = f(t)$ where t is developer's comments and T is the corresponding representation. In the Multi-Modal ACR task, each data example contains three parts: original code fragment, revised code fragment, and developer's comments. Thus, for each data, we can get three representations $\mathbf{C}^O, \mathbf{C}^R, \mathbf{T}^D$, corresponding to the original code fragment, revised code fragment, and the developer's comments.

2) Combined Representation: As shown in Fig. 2, after getting three representations, we first calculate the difference between the original code fragment and the revised code fragment:

$$\mathbf{C} = \mathbf{C}^O - \mathbf{C}^R \tag{5}$$

where $\mathbf{C}$ denotes the difference between the two code fragments. Then, we need to combine the difference representation with the comments representation, because the developer's comments describe the difference between the two code fragments.

$$\mathbf{r} = [\mathbf{C}, \mathbf{T}^D] \tag{6}$$

Thus, we get the combined representation for this data sample.

3) Prediction: In this part, we make the prediction according to the combined representation $\mathbf{r}$.

$$\mathbf{y} = \mathrm{softmax}(\mathbf{W}\mathbf{r} + \mathbf{b}) \tag{7}$$

where $\mathrm{softmax}(\cdot)$ is the softmax function to obtain the output distribution of the classifier, $\mathbf{W}$ and $\mathbf{b}$ are weights and biases, respectively.

The objective to train the classifiers is defined as minimizing the weighted cross entropy loss between the predicted and ground-truth distributions:

$$\mathcal{L} = -\sum_{i=1}^{S}(w^O y_i log\hat{p}_i + w^R(1-y_i)log(1-\hat{p}_i)) + \lambda \left\|\Theta\right\|_2 \tag{8}$$

where S is the number of training samples, w^O denotes the weight of incorrectly predicting a rejected change as approved, and w^R denotes the weight of incorrectly predicting an approved change as rejected. These two terms provide the opportunity to handle an imbalanced label distribution. λ is the weight of the L_2 regularization term. Θ denotes all trainable parameters.

IV. Experimental Design

This section introduces the process of the experiment, including the repository selection and the data construction, baseline setting, evaluation metrics, and experimental setting.

TABLE I: Statistics of the MACR dataset.

Repository	#samples	#rejected	reject rate
accumulo	20,903	9,042	43.3%
ambari	41,424	14,262	34.4%
beam	82,438	22,141	26.9%
cloudstack	37,599	18,703	49.7%
commons-lang	9,958	8,959	90.0%
flink	133,614	112,733	84.4%
hadoop	39,958	33,861	84.7%
incubator-pinot	13,903	2,118	15.2%
kafka	64,872	37,077	57.2%
lucene-solr	58,823	50,042	85.1%
shardingsphere	19,993	1,216	6.1%

A. Dataset Construction

We selected 11 projects from Github belonging to the Apache Foundation, which is a widely used code review source [2], [4]. Three of them (*accumulo, ambari, cloudstack*) were selected by Shi et al. [2]. The remaining projects (*beam, commons-lang, flink, hadoop, incubator-point, kafka, lucene-solr, shardingsphere*) were chosen because they have over 2000 stars. The language of all the projects is Java.

For data processing, we extracted all issues belonging to these projects from 2015 to 2020. Among these issues, many do not involve code submission, but only provide feedback, so we chose the types *PullRequestEvent* and *PullRequestReviewCommentEvent*.

In many practical cases, we can easily extract the original code, the revised code and the developer's comments from the issue, but because these codes are usually contained in many files, it is difficult for us to use them directly as the input of the network. Therefore, we assume that all of the changes are independent and identically distributed [2], so there is no connection between these changes, and if a file contains many changed methods, we can split these methods independently as inputs.

Further, if we add a new method or delete a whole method, half of the input data is empty. So we discard these data because they cannot be fed into the network. That is, we only consider the case where the code has been changed. In addition, considering that the submitted data may be too large, we subdivide the code submitted each time into the method level for subsequent experiments.

After processing the data, each piece of data comprises four parts: the original code fragment, the revised code fragment, the developer's comments and the label. The original code fragment and the revised code fragment are both method-level Java programs. The developer's comments are in English. The label uses 0 and 1 to represent rejection and acceptance. The basic statistics of the MACR are summarized in Table I.

In this paper, the rate of the rejection between 6% and 90% means that there is class imbalance during model training and it will lead to poor performance, so we use a weight-based class imbalance approach, setting the 'class_weight' parameter to 'balance' in our model.

B. Comparison Models

In this paper, we compared our proposed model CLMN with three other models in the Multi-Modal ACR task. These models use different methods (including delimiter-based method, token-based method, tree-based method) to obtain the code features. The baseline models are as follows:

- **DACE** [2] divides the code according to the delimiter and designs a pairwise autoencoder to compare the code.
- **TBRNN** serializes the AST into tokens and uses an RNN to capture the syntactic and semantic information.
- **ASTNN** [5] splits large ASTs into a sequence of small statement trees and calculates the representation distance to compare the code.

Please note that for all the methods above, we add the same text encoder **RoBERTa** to make all the methods can get the developer's comments as the input.

In order to ensure the fairness of the experiment and the stability of the results, we ran all the methods on the new MACR dataset, and each experiment was repeated 30 times.

C. Evaluation

Since the Multi-Modal ACR can be formulated as a binary classification problem (accept or reject) [2], we choose the F1-measure (F1) because it is widely used and the Matthews correlation coefficient (MCC) because it can better evaluate the performance of a model on imbalanced datasets.

The calculation formula of F1 is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

where TP, FP, FN, and TN represent True Positives, False Positives, False Negatives, and True Negatives, respectively. The F1 score is the harmonic mean of the precision and recall. The value range of F1 is [0,1].

The calculation formula of MCC is:

$$MCC = \frac{TP \times TN - TP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{12}$$

The value range of MCC is [-1,1]. For all metrics, higher values are better.

D. Experimental Setting

In our experiments, we used the javalang tools[2] to obtain ASTs for Java code, and we trained embeddings of symbols using word2vec with the Skip-gram algorithm. For the code encoder, the embedding size was set to 512. The hidden size of Bi-GRU was 512. The number of GCN layers was 4. For the text encoder, we used the default parameters. The coefficients w^O and w^R were related to the dataset, and the coefficient λ of L_2 regularization item was set to 10^{-5}. Adam was utilized as

[2]https://github.com/c2nes/javalang

TABLE II: Performance comparison in terms of F1.

Repository	DACE	TBRNN	ASTNN	CLMN
accumulo	0.995	0.99	0.995	**0.996**
ambari	0.977	0.979	0.979	**0.98**
beam	0.995	0.994	0.995	**0.996**
cloudstack	0.979	0.979	0.98	**0.981**
commons-lang	0.957	0.963	**0.965**	0.964
flink	0.421	0.959	0.972	**0.979**
hadoop	0.947	0.934	0.949	**0.952**
incubator-pinot	0.859	0.795	0.954	**0.994**
kafka	0.985	0.691	0.985	**0.986**
lucene-solr	0.978	**0.979**	0.978	0.975
shardingsphere	0.992	0.991	0.927	**0.994**
Average	0.917	0.932	0.971	**0.981**

TABLE III: Performance comparison in terms of MCC.

Repository	DACE	TBRNN	ASTNN	CLMN
accumulo	0.989	0.978	0.99	**0.991**
ambari	0.936	0.941	0.943	**0.946**
beam	0.982	0.977	0.983	**0.985**
cloudstack	0.957	0.957	0.96	**0.961**
commons-lang	0.954	0.96	**0.962**	0.961
flink	0.306	0.952	0.967	**0.975**
hadoop	0.939	0.923	0.94	**0.944**
incubator-pinot	0.466	0.146	0.894	**0.961**
kafka	0.974	0.457	0.974	**0.975**
lucene-solr	0.975	**0.976**	0.975	0.971
shardingsphere	0.876	0.863	0.295	**0.899**
Average	0.85	0.83	0.898	**0.961**

the optimizer with a learning rate of 10^{-5} to train the model, and the mini-batch was 64. We random initialized all the $\mathbf{W}$ and $\mathbf{b}$ with a uniform distribution.

All the experiments were conducted on a server with 24 cores of 3.8GHz CPU and a GeForce RTX 3090 GPU.

V. Experimental Results

This section shows the performance of our proposed method CLMN with other baseline methods. Therefore, we put forward the following research questions:

- RQ1: Does our proposed model CLMN outperform other models for Multi-Modal ACR?
- RQ2: Does the addition of developer's comments improve the performance of ACR?
- RQ3: How about the impact of contrastive learning on model robustness?

A. Does our proposed model CLMN outperform other models for Multi-Modal ACR?

Tables II, III show the comparison results on the MACR dataset. The results show that the proposed CLMN consistently outperforms all comparison models. This verifies the effectiveness of our proposed method at Multi-Modal ACR.

Compared with DACE, our model CLMN achieves the best performance in both the F1 and MCC metrics, with 6.4% and 11.1% improvement, respectively. Compared with TBRNN and ASTNN, CLMN improved by 4.9% and 1% respectively in F1. In the MCC metric, CLMN also gained 13.1% and 6.3% improvement.

TABLE IV: Impact of the addition of developer comments.

Repository	CLMN w/o Text		CLMN	
	F1	MCC	F1	MCC
accumulo	0.761	0.448	**0.996**	**0.991**
ambari	0.704	0.013	**0.98**	**0.946**
beam	0.824	0.321	**0.996**	**0.985**
cloudstack	0.532	0.215	**0.981**	**0.961**
commons-lang	0.469	0.474	**0.964**	**0.961**
flink	0.367	0.237	**0.979**	**0.975**
hadoop	0.393	0.298	**0.952**	**0.944**
incubator-pinot	0.673	0.181	**0.994**	**0.961**
kafka	0.604	0.298	**0.986**	**0.975**
lucene-solr	0.392	0.297	**0.975**	**0.971**
shardingsphere	0.839	0.191	**0.994**	**0.899**
Average	0.596	0.27	**0.981**	**0.961**

* CLMN w/o Text means the CLMN without the Text Encoder.

TABLE V: Impact of contrastive learning on model robustness.

Repository	*beam* based CLMN		*self* based CLMN	
	F1	MCC	F1	MCC
accumulo	0.996	0.99	0.996	0.991
ambari	0.98	0.946	0.98	0.946
cloudstack	0.98	0.961	0.981	0.961
commons-lang	0.948	0.944	0.964	0.961
flink	0.973	0.968	0.979	0.975
hadoop	0.951	0.943	0.952	0.944
incubator-pinot	0.989	0.934	0.994	0.961
kafka	0.984	0.973	0.986	0.975
lucene-solr	0.973	0.968	0.975	0.971
shardingsphere	0.993	0.887	0.994	0.899
Average	0.977	0.951	0.98	0.958

* *beam* based CLMN: using the Code Encoder pre-trained by *beam*.
* *self* based CLMN: using the Code Encoder pre-trained by themselves.

One aspect of our improvement comes from using a better code encoder. Our code encoder uses the Simplified AST as input, which is better than the source code fragment and the original AST. Simplified AST has fewer tokens and tighter node connections. Besides, our code encoder uses the GCN layers to get more syntactic and semantic information than the Bi-GRU layers in other methods.

Another aspect of our improvement comes from the application of contrastive learning. We utilize contrastive learning to obtain a more evenly distributed representation of the source code, which allows our model to perform better when encountering few-shot and 0-shot samples. In addition, the pre-training parameters obtained by contrastive learning can be flexibly applied to other tasks (discussed in RQ3), thereby reducing the consumption of resources.

In summary, this is the first study to apply contrastive learning to improve the performance in the multi-modal automatic code review task. Our experiments show the effectiveness of

exploiting the Simplified AST and the contrastive learning.

B. Does the addition of developer's comments improve the performance of ACR?

Table IV shows the comparison results between the CLMN with and without the developer's comments. The results show that CLMN without the developer's comments tends to drop sharply.

We can observe that the CLMN without the developer's comments has the worst result on *ambari*, with a 93% drop in MCC compared to the original result. On the *flink* project, the F1 value of CLMN without Text has dropped 61%. On average, CLMN without Text has an overall decrease of 69% in MCC and 38% in F1. This establishes that Multi-Modal ACR can make better decisions with the developer's comments rather than only using the code fragments.

C. How about the impact of contrastive learning on model robustness?

To investigate the robustness of parameters obtained by contrastive learning, we use *beam* as the original pre-training project. After getting the pre-trained parameters of the code encoder of *beam*, we apply it to the remaining projects with fine-tuning to obtain the performance in other projects. Table V shows the impact of contrastive learning on model robustness.

We can find that after using *beam's* pre-trained parameters, the performance of our model in the other ten repositories does not drop significantly compared to the performance pre-trained by themselves. There is even no drop in some repositories. On average, there is only a drop of 0.3% in F1 and 0.7% in MCC. In other words, the hyperparameters obtained using contrastive learning can be easily extended to different projects and obtain similar results. When the model encounters an unseen code fragment (few-shot or 0-shot), it can also generate its vector representation accurately.

The results are soundproof that contrastive learning can effectively increase the robustness of the model. It also demonstrates that the parameters obtained by contrastive learning can help the code get a more uniform distribution in the vector space, resulting in better performance across different projects.

VI. Conclusion

In this paper, we first present MACR, which is the first open-sourced Multi-Modal dataset for Multi-Modal ACR. Then, we propose a novel model called CLMN. CLMN has two encoders, SimAST-GCN as the code encoder and RoBERTa as the text encoder. The two encoders are separately pre-trained using contrastive learning to obtain a uniformly distributed representation, which improves the robustness of the model and solves the few-shot and 0-shot issues. Finally, we combine the two encoders to obtain a vector representation of the code review and predict the results. Experimental results on the MACR dataset show that our proposed model CLMN outperforms state-of-the-art methods. Our code and experimental data are publicly available at https://github.com/SimAST-GCN/CLMN.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (61772263, 61872177, 61972289, 61832009), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, "Modern code review: a case study at google," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 181–190.

[2] S.-T. Shi, M. Li, D. Lo, F. Thung, and X. Huo, "Automatic code review by learning the revision of source code," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33(01), 2019, pp. 4910–4917.

[3] J. K. Siow, C. Gao, L. Fan, S. Chen, and Y. Liu, "Core: Automating review recommendation for code changes," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 284–295.

[4] B. Wu, B. Liang, and X. Zhang, "Turn tree into graph: Automatic code review via simplified ast driven graph convolutional network," *arXiv preprint arXiv:2104.08821*, 2022.

[5] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," pp. 783–794, 2019.

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[7] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 712–721.

[8] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K.-i. Matsumoto, "Who should review my code? a file location-based code-reviewer recommendation approach for modern code review," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2015, pp. 141–150.

[9] Z. Xia, H. Sun, J. Jiang, X. Wang, and X. Liu, "A hybrid approach to code reviewer recommendation with collaborative filtering," in *2017 6th International Workshop on Software Mining (SoftwareMining)*. IEEE, 2017, pp. 24–31.

[10] V. Balachandran, "Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 931–940.

[11] G. Díaz and J. R. Bermejo, "Static analysis of source code security: Assessment of tools against samate tests," *Information and software technology*, vol. 55, no. 8, pp. 1462–1476, 2013.

[12] L. Mou, G. Li, Z. Jin, L. Zhang, and T. Wang, "Tbcnn: A tree-based convolutional neural network for programming language processing," *arXiv preprint arXiv:1409.5718*, 2014.

[13] T. Shippey, D. Bowes, and T. Hall, "Automatically identifying code features for software defect prediction: Using ast n-grams," *Information and Software Technology*, vol. 106, pp. 142–160, 2019.

[14] C. Zhang, Q. Li, and D. Song, "Aspect-based sentiment classification with aspect-specific graph convolutional networks," *arXiv preprint arXiv:1909.03477*, 2019.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[16] T. Chen, Y. Sun, Y. Shi, and L. Hong, "On sampling strategies for neural network-based collaborative filtering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 767–776. [Online]. Available: https://doi.org/10.1145/3097983.3098202

[17] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

Utilizing Edge Attention in Graph-Based Code Search

Wei Zhao
School of Software Engineering
Tongji University
Shanghai, China
2031569@tongji.edu.cn

Yan Liu*
School of Software Engineering
Tongji University
Shanghai, China
yanliu.sse@tongji.edu.cn

***Abstract*—Code search refers to searching code snippets with specific functions in a large codebase according to natural language description. Classic code search approaches, using information retrieval technologies, fail to utilize code semantics and bring noisy and irrelevant keywords. During the last recent years, there has been an ample increase in the number of deep learning-based approaches, which embeds lexical semantics into unified vectors to achieve higher-level mapping between natural language queries and source code. However, these approaches are struggling with how to mine and utilize deep code semantics. In this work, we study how to leverage deeper source code semantics in graph-based source code search, given graph-based representation is a promising way of capturing program knowledge and has rich explainability. We propose a novel code search approach called EAGCS (Edge Attention-based Graph Code Search), which is composed of a novel code graph representation method called APDG (Advanced Program Dependence Graph) and a graph neural network called EAGGNN (Edge Attention-based GGNN) which can learn the latent code semantics of APDG. Experiment results demonstrate that our model outperforms the GGNN-based search model and DeepCS. Moreover, our comparison study shows that different edge enhancement strategies have different contributions to learning the code semantics.**

Keywords—semantic code search; graph neural network; graph representation learning

I. INTRODUCTION

Code search is one of the most common activities in software development, some studies [1][2] have shown that 19% of developers' time will be spent searching desirable code snippets. Especially in recent years, with the rise of agile development [3], the developer needs to iterate the projects rapidly. Accurate search results can be reused with only slight reconstruction and help quickly realize specific project functions to boost developers' productivity. However, the existing code search engines, such as those on GitHub [4] and Stack Overflow [5], treat the source code as plain text and search the code snippets mainly based on keyword matching, lacking the semantic understanding of natural language and source code. Therefore, it is difficult to use such IR (Information Retrieve) techniques to optimize code search.

More recently, deep learning-based code search can reduce the interference of noisy keywords and learn code features by vectoring the code, which can recognize semantically related words. For example, DeepCS [6] obtained the API sequence, method name and token information according to defined rules and embedded them into unified vectors to represent the code. Therefore, the basic code representation method has a far-reaching impact on the expression of code semantics, which will further affect search performance. As code is structural and has unique language-specific semantics, the graph is a natural and effective representation of code. Taking Java as an example, there are a series of implicit relationships between code elements [7], such as the order of method calls, class inheritance, etc., and these relationships can reveal the potential code semantics. Inspired by this, many researchers utilized variants of abstract syntax tree (AST) [8] and other code graphs to represent latent code semantics. For instance, DeepCS extracted API sequence from AST, and Xiang et al. [9] generated a code graph based on AST with different nodes (terminal/non-terminal nodes) and edges. However, code with different syntax structures may express the same functionality [10], as shown in Figure 1 and Figure 2, which indicates that simply using AST to represent code is not enough to accurately express the deep code semantics. So we need to break through the current limitations of sequential data and enhance code semantics via utilizing the rich structural information behind programs.

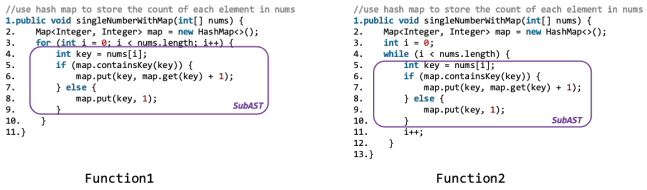


Figure 1. Two code snippets with the same functionality

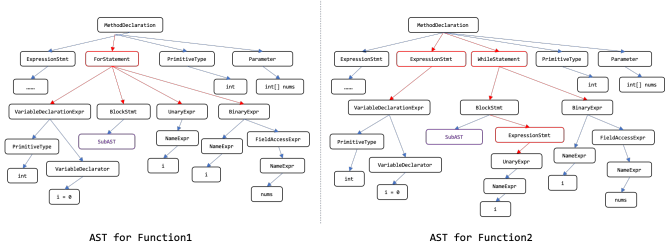


Figure 2. The two ASTs correspond to the code snippets in Figure 1, where the nodes and edges marked in red indicate the structural differences.

To deal with the aforementioned challenges and utilize structural information, we propose a novel code search approach called EAGCS, which can significantly enhance the expression of structural and semantic information of source code. More specifically, we first construct a statement-level advanced

* Corresponding Author

DOI reference number: 10.18293/SEKE2022-078

program dependence graph (APDG) which transforms three common control statements and adds control and data edges to improve the awareness of neighbors. Program dependence graph (PDG) [11] is the graphical representation of a program where nodes represent program statements and edges represent latent dependence information, but it fails to reflect the order in which statements are executed and the implicit control logic [12]. In APDG, we enrich code semantics based on our defined data and control dependence rules, introducing the statement execution information and control logic missed in PDG. Besides, the APDG is constructed based on AST, which can keep the syntax information of source code, and statement-level graph nodes can preserve local semantics compared with excessive fine granularity nodes in AST (i.e., *NameExpr* and *ExpressionStmt* nodes in Figure 2). Concentrating on multiple edge types, we apply EAGGNN to learn the semantic features of ADPG. Furthermore, we calculate the cosine similarity of the code and description vector embedded by our model and search the top-k relevant code snippets according to the given user queries. Experiments have been conducted and the results demonstrate that our model outperforms the other start-of-the-art models.

The main contributions of this paper are as follows:

- We introduce a novel statement-level code representation method called APDG, which optimizes the traditional PDG and strengthens both data and control dependence information to enrich the edge semantics.
- We propose EAGCS, a graph-based code search approach that enhances the GGNN [13] via an edge attention mechanism to improve the expression of code semantics in APDG.
- We conduct experiments on our model and other start-of-the-art models and the results have shown that our model outperforms the others. Besides, we also explore the impact of different edges on the model performance.

II. Related Work

With the in-depth study of code search in recent years, a variety of research methods have been proposed. Traditional code search methods treated source code as plain text and obtained the most relevant code snippets through the information retrieve (IR) technology. For example, Lv et al. [14] designed CodeHow, which expanded the user query with the APIs and applied an extended boolean model to perform code search. While Portfolio [15] combined keyword matching and PageRank to retrieve a series of functions according to user input.

To solve the problem that code snippets without keywords related to the description cannot be searched in the above-mentioned models, Gu et al. [6] proposed the first deep learning-based code search tool named DeepCS. DeepCS embedded code snippets and natural language descriptions into high-dimensional vector space separately, which can recognize semantically related words. On this foundation, some other previous works used an attention mechanism or reconstructed the model structure to boost the search performance. Shuai et al. [16] utilized CARLCS-CNN based on Convolutional Neural Network (CNN) instead of LSTM used in DeepCS and built a semantical correlation between the code and description vectors via a co-attention mechanism. The work in [17] applied a self-attention network to learn the contextual representation and global semantic relations for code snippets and their corresponding queries.

Some other researchers are mainly dedicated to enhancing the representative ability of code semantics. Wan et al. [18] constructed the code features with the sequential tokens, ASTs (abstract syntax trees), and CFGs (control-flow graphs) to represent syntactic and semantic information of source code. Similarly, Zeng et al. [10] encoded source code into variable-based flow graphs and utilized an improved gated graph neural network (GGNN) to model more precise code semantics. Liu et al. [19] transformed code snippets and descriptions into ASTs and dependence parsing graphs separately to capture their joint semantic relationship.

In addition to semantic enhancement of source code, some work focused on query expansion and reinforcement. For instance, Sirres et al. [20] augmented user query with program elements, such as method and class names, from the extracted snippets. Xuan et al. [21] proposed DERECS to reinforce the code based on the method call and the structural characteristics of the code fragment, which reduced the difference between source code and query.

III. EAGCS

A. Overview

Figure 3 shows the overall structure of EAGCS, including 4 components: preprocessing, code graph generation, description embedding, and code graph embedding. Despite AST can reflect the syntax information of the source code, it is complex so we need to prune it to remove redundant nodes. The APDG we proposed not only retains the syntax information of AST in statement level which reduces the scale of code graph, but also adds data and control edges to enhance the code semantics. When the model searches code snippets, code semantics are explicitly expressed through multiple graph edges in APDG. Moreover, the edge attention-based GGNN (EAGGNN) can help to obtain a deeper understanding of APDG, it learns the node embeddings from multiple edges to focus on both data and control dependence. After embedding both descriptions and code snippets into unified vectors, the model can recommend code snippets with higher cosine similarity according to the given user query.

B. Preprocessing

1) Data Collection: To guarantee the performance of our model, we need a large-scale dataset that contains query descriptions and their corresponding code snippets. The dataset provided by DeepCS contains more than 18 million data items, which is used in most existing studies and is our ideal dataset. Unfortunately, the code graph generation approach we designed needs to be applied through the source code (raw data)， but the dataset provided by DeepCS has been already preprocessed and embedded, which fails to provide in-depth semantic information. Therefore we choose the dataset published by CodeSearchNet[1], which is a little less than that of DeepCS but

[1]https://github.com/github/CodeSearchNet

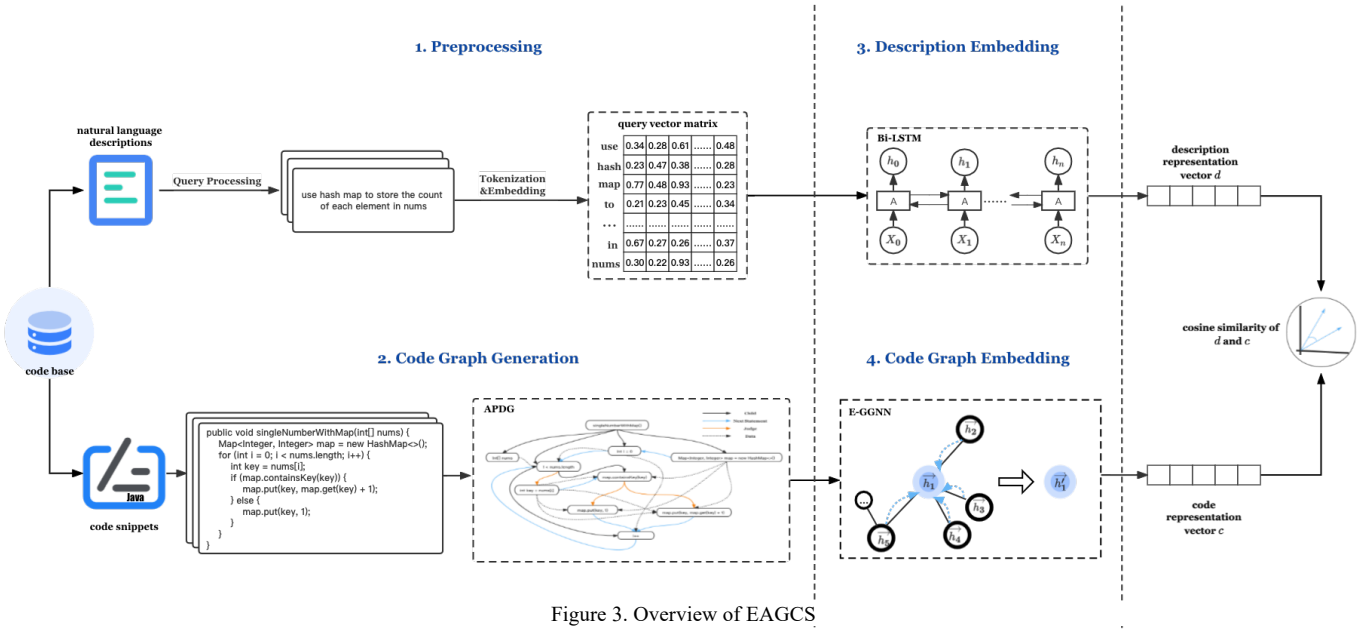


Figure 3. Overview of EAGCS

contains raw code snippets.

2) Data Processing: The CodeSearchNet dataset includes code snippets extracted from real projects. However, due to the complexity of the actual development process, participants, and project types, the raw data contains noisy and dirty data items. So we need to go through several processes to improve the quality of data fed to the model.

Query Processing: We take the comments corresponding to the code snippets as the user query and handle them according to the following flow.

- Segment the comment according to the "." character, and select the first sentence as a user query. (User comment may include multiple sentences, but the first sentence is usually complete enough to describe the code, and the following statements are only for supplementary explanation)
- Remove non-English symbols and stop words[1].

Code processing: we apply the *javaparser*[2] library to transform the code snippets and generate APDG based on our code graph generation approach.

- Convert the code snippets into class. (The original code fragment is at the function level)
- Transform the code snippet into APDG.
- Delete non-English characters and generate words according to lower CamelCase for each statement-level node.

Tokenization: We first count the frequency of words in query and node statements separately and build two dictionaries based on their top 10000 words. Furthermore, we represent each word with a unique numeric ID and transform the query statements and node statements into sequences of numerical tokens.

Embedding: Embedding is a technology that maps an object (i.e., a word) into a real vector, which can make objects with similar meanings have vectors with similar distances. The commonly used word embedding approaches are CBOW [22] and Skip-Gram [23]. In our work, we use the *nn.Embedding*[3] function of *pytorch*[4] to initialize the word embeddings and then retrieve them using indices.

C. Code Graph Generation Approach

Program dependence graph (PDG) is one of the most widely used directed code graphs [11], where the nodes represent program statements and the edges represent the interdependence between program statements, but it fails to reflect the order in which statements are executed [12]. In our work, we propose advanced PDG (APDG) which adds *NextStatement* edges in PDG and we also provide clear guidelines for the optimization of common control statements to explore more program semantics. More specifically, we consider 12 types of nodes: *Method Declaration*, *Parameter*, *Unary Expression*, *Variable Declaration Expression*, *Method Call Expression*, *Assign Expression*, *Construction Declaration*, *Try Statement*, *Class or Interface Declaration*, *Condition*, *Return Statement* and *Assert Statement* based on the AST and the Soot's [24] internal representation. Moreover, we have also defined the extraction rules of control dependence (*Child, NextStatement*, *Judge* Edges) and data dependence, which will be formally described in detail:

1) Control Dependence: control dependence defines the constraint relationship of statement execution, which can reflect the syntax, execution order, and control information.

Child: *Child* edges connect parent and child nodes in AST and point from parent node to child node which can reflect the control dependence of statements at the syntactic level.

NextStatement: *NextStatement* edges concatenate statements inside a block according to the context, indicating the order of statement execution. The dashed box represents the virtual structure for better illustration, which will not appear in real APDG.

[1]https://www.textfixer.com/tutorials/common-english-words.txt
[2]https://github.com/javaparser/javaparser
[3]https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html
[4]https://pytorch.org/

Judge: We transform three common control statements in AST: *IfStatement*, *ForStatement, WhileStatement* and add *Judge* edges to uncover the control logic.

2) Data Dependence: data dependence defines the constraint relationship of variables between statements. To mine the data dependence relationship, we have to keep a record of all accesses of all variables. The data dependence rule of variable *v* from statements s_1 to s_2 can be described as:

- *v* is defined or assigned in statement s_1.
- *v* is used in statement s_2.
- The scope of s_2 is inside the scope of s_1.
- If s_1 and s_2 have the same scope level, a *NextStatement* path exists from s_1 to s_2.

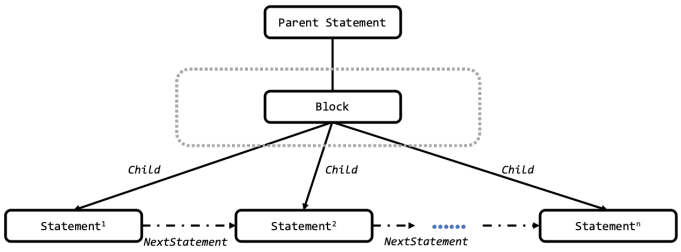


Figure 4. Illustration of Child and NextStatement edges.

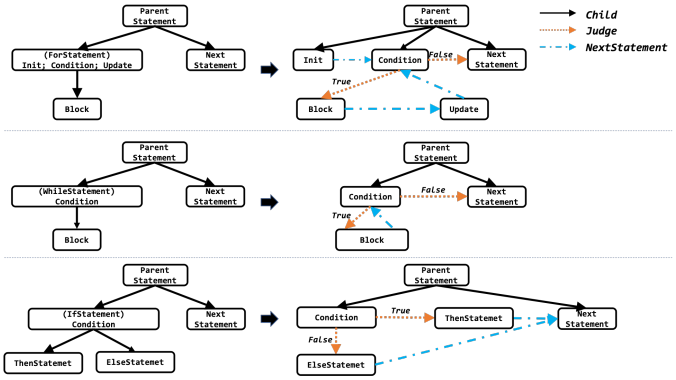


Figure 5. Illustration of control statements optimization and Judge edges.

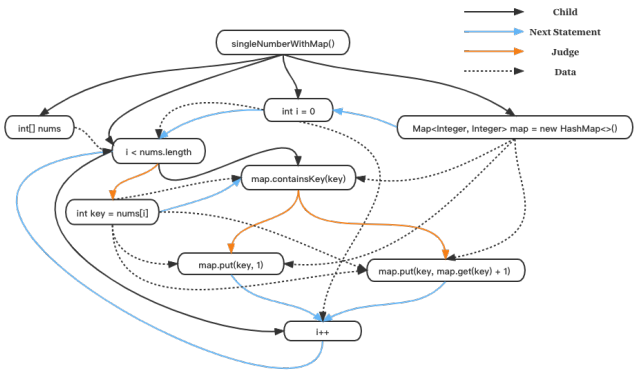


Figure 6. APDG corresponds to the code snippets in Figure 1.

As shown in Figure 6, we constructed APDG for the code snippets in Figure 1 based on the designed control dependence and data dependence rules. And latent code semantics can be expressed through node contents and multiple edges.

D. Description Embedding

By word embedding, we view a description $\boldsymbol{D}$ as a sequence of token vectors: $\boldsymbol{w_1}$, ..., $\boldsymbol{w_N}$, $\boldsymbol{D} = \{\boldsymbol{w_1}, ..., \boldsymbol{w_N}\}$. We use a bi-LSTM to extract semantic information from the input in both forward and reverse directions, and embed the description into a vector $\boldsymbol{d}$.

$$\overrightarrow{\boldsymbol{h_t}} = \overrightarrow{LSTM}\left(\boldsymbol{w_t}, \overrightarrow{\boldsymbol{h_{t-1}}}\right) \quad (1)$$

$$\overleftarrow{\boldsymbol{h_t}} = \overleftarrow{LSTM}\left(\boldsymbol{w_t}, \overleftarrow{\boldsymbol{h_{t+1}}}\right) \quad (2)$$

$$\boldsymbol{h_t} = \overrightarrow{\boldsymbol{h_t}} \oplus \overleftarrow{\boldsymbol{h_t}} \ \forall t = 1, \ldots, N \quad (3)$$

$$\boldsymbol{d} = \boldsymbol{maxpooling}(\boldsymbol{h_1}, \ldots, \boldsymbol{h_N}) \quad (4)$$

where $\boldsymbol{w_t} \in \mathbb{R}^d$, $\boldsymbol{h_t}$ is the hidden states at step *t*, $t = 1, ..., N$, *N* is the length of the sequence, $\oplus$ is the concatenation operation.

E. Code Graph Embedding

1) Node Embedding: We view a graph node $\boldsymbol{V}$ as several token vectors: $\boldsymbol{t_1}$, ...,$\boldsymbol{t_M}$. $\boldsymbol{V} = \{\boldsymbol{t_1}, ..., \boldsymbol{t_M}\}$. For there is no strict order between these tokens, we use a multilayer perceptron (MLP) to embed the node into a vector $\boldsymbol{n}$.

$$\boldsymbol{h_i} = \tanh(W^T \boldsymbol{t_i}) \ \forall i = 1, \ldots, M \quad (5)$$

$$\boldsymbol{n} = maxpooling([\boldsymbol{h_1}, \ldots, \boldsymbol{h_M}]) \quad (6)$$

where $\boldsymbol{t_i} \in \mathbb{R}^d$, $\boldsymbol{i} = 1, ..., M$, *M* is the number of the tokens, $\boldsymbol{W^T}$ is the learnable matrix in the MLP.

2) Edge Attention Based GGNN

Gated Graph neural network (GGNN) is a kind of graph neural network (GNN) that directly uses graph data as the structured input. For most GNNs, much information sharing might reduce the weight of the original information of the node itself, which can lead to overfitting. To overcome this issue, GGNN can selectively remember the hidden information of neighbor nodes and the hidden information in the process of node iteration by adding a GRU [13] component. As we have already enriched edge semantics in APDG, we propose EAGGNN which enhances the GGNN via an edge attention mechanism to deal with different types of edges to focus on both data and control dependence information for each iteration. Considering the program graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is the node collection and $\mathcal{E}$ is the adjacency matrix. For each node $\boldsymbol{v} \in \mathcal{V}$, $\boldsymbol{h_v^{(0)}}$ is the initial hidden state of node $\boldsymbol{v}$ through node embedding, and $\boldsymbol{h_v^{(k)}}$ is the hidden state of node $\boldsymbol{v}$ in hop $\boldsymbol{k}$.

$$\boldsymbol{a_{v|D}^{(k)}} = A_{v|D}^{\tau}\left[\boldsymbol{h_1^{(k-1)\tau}} \ldots \boldsymbol{h_{|\mathcal{V}|}^{(k-1)\tau}}\right] + \boldsymbol{b} \quad (7)$$

$$\boldsymbol{a_{v|C}^{(k)}} = A_{v|C}^{\tau}\left[\boldsymbol{h_1^{(k-1)\tau}} \ldots \boldsymbol{h_{|\mathcal{V}|}^{(k-1)\tau}}\right] + \boldsymbol{b} \quad (8)$$

$$\boldsymbol{a_v^{(k)}} = \boldsymbol{a_{v|D}^{(k)}} \oplus \boldsymbol{a_{v|C}^{(k)}} \quad (9)$$

$$\boldsymbol{z_v^{(k)}} = \sigma\left(W^z \boldsymbol{a_v^{(k)}} + U^z \boldsymbol{h_v^{(k-1)}}\right) \quad (10)$$

$$\boldsymbol{r_v^{(k)}} = \sigma\left(W^r \boldsymbol{a_v^{(k)}} + U^r \boldsymbol{h_v^{(k-1)}}\right) \quad (11)$$

$$\widetilde{\boldsymbol{h_v^{(k)}}} = \tanh\left(W \boldsymbol{a_v^{(k)}} + U\left(\boldsymbol{r_v^{(k)}} \odot \boldsymbol{h_v^{(k-1)}}\right)\right) \quad (12)$$

$$\boldsymbol{h_v^{(k)}} = \left(1 - \boldsymbol{z_v^{(k)}} \odot \boldsymbol{h_v^{(k-1)}} + \boldsymbol{z_v^{(k)}} \odot \widetilde{\boldsymbol{h_v^{(k)}}}\right) \quad (13)$$

$$\boldsymbol{c} = maxpooling\left(\boldsymbol{h_1^K} \ldots \boldsymbol{h_{|\mathcal{V}|}^K}\right) \quad (14)$$

where $A_{v|D}$ and $A_{v|C}$ are the columns corresponding to node $\boldsymbol{v}$ in data adjacency matrix and control adjacency matrix

separately, $z_v^{(k)}$ is the update gate, $r_v^{(k)}$ is the reset gate, K is the number of hops in EAGGNN and $\boldsymbol{c}$ is the final code representation.

F. Model Training

Considering a code-description pair $P(c, d^+)$, where $c \in C$, $d^+ \in D$, C denotes the set of code snippets, D denotes the set of descriptions, c denotes the single code snippet, d^+ denotes the corresponding query description of c and d^- denotes another randomly selected query description from D, $d^- \in D$, $d^- \neq d^+$. Then we rebuild the pair P as $P' = (c, d^+, d^-)$ and train the model by minimizing the loss function $L(\theta)$ that is formulated as:

$$L(\theta) = \sum_{(c,d^+,d^-)\in P'} \max(0, \beta - \cos(c, d^+) + \cos(c, d^-)) \quad (15)$$

where θ denotes the model parameters, d^+denotes the positive description, d^-denotes the negative description, *cos* is cosine similarity function and β denotes the constant margin.

IV. Experiments

A. Dataset

We choose the dataset of CodeSearchNet as the training set which contains 454,451 samples, and then filter these samples according to the following rules:

- Remove duplicate queries. (The dataset contains override or overload functions that have the same comments, and we only choose the sample that appears first)
- Remove code snippets that cannot be compiled properly
- P with description d that contains less than 3 words or more than 20 words will be filtered out. (The excessively long query length does not meet the actual user requirements)
- P with code c that is less than 3 lines and more than 20 lines will be filtered out. (Too short code snippets are meaningless, while too long code is difficult to understand)

As result, we obtained 126,363 pieces of data and converted all the processed code snippets into ASTs and APDGs. In addition, statistics on the maximal/average/minimal number of nodes and edges of these ASTs and APDGs were conducted and the results were shown in Figure 7, which indicated that APDG effectively reduced the complexity of code graph and was conducive to model training.

For the test set, we selected the 10,000 code-query pairs provided by Gu et al. [6]. Through the same filtering flow, we gained 4,548 pieces of data for evaluation and utilized an automatic evaluation approach which used the 4,548 queries as model inputs while the corresponding code snippets were treated as ground truth. In the automatic evaluation, we randomly selected 99 other code snippets and combined them with the ground truth as the search base for a query input. This automatic evaluation approach can avoid the bias caused by manual ranking. Besides, selecting training and test sets from different projects can examine the generalization ability of the model.

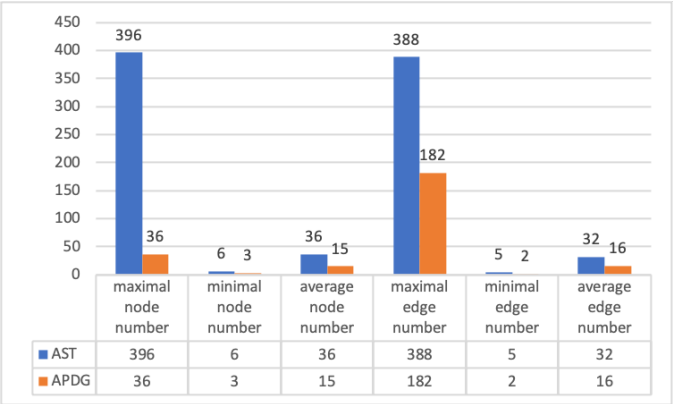


Figure 7. Statistical results of APDGs and ASTs on CodeSearchNet.

B. Evaluation Metrics

SuccessRate@k: The SuccessRate@k measures the percentage of queries for which the corresponding ground-truth code snippet could exist in the top k ranked results and it can be formulated as:

$$SuccessRate@k = \frac{1}{|Q|}\sum_{q=1}^{|Q|} \delta(FRank_q \leq k) \quad (16)$$

where $\boldsymbol{Q}$ denotes the set of queries, $\delta(\cdot)$ denotes the function that returns 1 if the input is true and 0 otherwise, Frank denotes the rank position of the ground truth in the result list.

MRR: MRR is the average of the reciprocal ranks of results of a query set $\boldsymbol{Q}$, which can be defined as follows:

$$MRR = \frac{1}{|Q|}\sum_{q=1}^{|Q|} \frac{1}{Frank_q} \quad (17)$$

where the reciprocal ranks is the inverse of Frank. As we expect to find the ground truth in the top 10 results, we set $1/Frank_q$ to 0 if $FRank_q$ is larger than 10.

C. Baseline Models

DeepCS is the state-of-the-art neural network to retrieve relevant code snippets given a query description. It extracted the method name, API sequence, and tokens of a method to represent the code semantics, and then embedded code and description to get the unified vectors to calculate the similarity between them.

GGNN represents the gated neural network without edge attention based on our APDG. It utilized the GGNN on data and control dependence separately and then combined the two hidden states via an *avgpooling* function to represent the code.

D. Implementation Details

We embedded the top 10000 tokens for the code statements and descriptions separately with a 128-dimensional size and used Adam [25] as the optimizer. The model was trained for 250 epochs while the batch size was set as 100. The iteration times of EAGGNN were set as 4 and the dropout was set as 0.6 in the word embedding layer for the learning process.

E. Results

Table I summarizes the experiment results of our model and the baseline models on the test set. The R@1, R@5, and R@10 denote the results of SuccessRate@k, where k is 1, 5, 10. The results have shown that our model outperforms the state-of-the-art models. The R@1, R@5, R@10, and MRR of EAGCS are respectively 0.15, 0.16, 0.18, and 0.15 higher than GGNN, which indicates the edge attention mechanism can effectively aggregate the edge information.

TABLE I. COMPARISON OF THE MODEL PERFORMANCE BETWEEN OUR MODEL AND THE BASELINE MODEL

Model	R@1	R@5	R@10	MRR
DeepCS	0.2199	0.3628	0.4574	0.2846
GGNN	0.4268	0.5500	0.5910	0.4826
EAGCS	**0.5785**	**0.7071**	**0.7682**	**0.6357**

TABLE II. EFFECT OF EACH EDGE TYPE

Model	D	C	R@1	R@5	R@10	MRR
EAGCS	✓		0.3173	0.4807	0.5706	0.3894
		✓	0.2718	0.4576	0.5521	0.3507
	✓	✓	**0.5785**	**0.7071**	**0.7682**	**0.6357**

Table II presents the influence of different types of edge on search performance. The header D and C indicate whether data edge and control edge exist in APDG, where the checkmark represents that the corresponding edge is added. Incorporating both data and control dependence can express the code semantics to the greatest extent and can get the best model performance. Results also show that data dependence has a slightly greater weight than control dependence for that all metrics are higher. Data dependence reflect the flow of variables between basic blocks under the control structure, which is a further and deeper analysis of program features.

F. Threats to Validity

Our work may suffer from four validity. The first one is the model re-implementation. Replicating the baseline models may introduce some errors. To mitigate this threat, we used the authors' open-source projects on GitHub and processed our dataset into the same format required by the projects. The second one is the selected dataset. Because the dataset provided by DeepCS is vectorized, we can't obtain the original code snippets for our model to generate code graphs, so we utilized the CodeSearchNet dataset for model training, which was smaller than that of DeepCS but contained raw code snippets. Furthermore, the training and test set shared the same preprocessing flow. The third one is the model evaluation. We took the automatic evaluation approach to avoid manual risks. Given an input query, we set the same search base for all baseline models. The experiment results may be influenced by the scale or the origin of the search base, which is our future research content. The last one is the model comparision. In our experiment, We applied the same dataset, ran all the models in the same hardware environment, and adopted the same data preprocessing process to reduce this threat. While DeepCS does not perform on the graph structure, more related baselines may be needed to justify the advantages introduced by our proposed model in the future.

V. CONCLUSION

How to accurately understand and express code semantics has become a key challenge in the process of code search. In this paper, we propose a novel graph-based code search method called EAGCS, which mines latent code semantics by enhancing edge information in APDG and embeds the APDG into graph-level vector via edge attention-based GGNN to boost the semantic expression. In the future, we will strive to optimize the code graph structure and model architecture to improve search performance. We also plan to investigate the influence of the number of nodes in APDG and the length of query statements on the search results. Furthermore, how to excavate potential user search intention and reinforce user query is another rich field worthy to be penetratingly explored.

REFERENCES

[1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 1589-1598.

[2] K. Kevic and T. Fritz, "Automatic search term identification for change tasks," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 468-471.

[3] S. Nerur and V. G. Balijepally. "Theoretical reflections on agile development methodologies". Communications of the ACM, 2007, pp. 79-83.

[4] 2022. Github. Retrieved February 14, 2022 from https://github.com.

[5] 2022. Stack Overflow. Retrieved February 14, 2022 from https://stackoverflow.com.

[6] X. Gu, H. Zhang, and S. Kim, "Deep Code Search," *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 933-944, doi: 10.1145/3180155.3180167.

[7] T. Ben-Nun, A. S. Jakobovits, and T. Hoefler, "Neural code comprehension: A learnable representation of code semantics," in *Advances in Neural Information Processing Systems*, 2018, pp. 31.

[8] I. Neamtiu, J. S. Foster, and M. Hicks, "Understanding source code evolution using abstract syntax tree matching," in *Proceedings of the 2005 international workshop on Mining software repositories*, 2005, pp. 1-5.

[9] X. Ling, L. Wu, S. Wang, G. Pan, T. Ma, F. Xu, A. X. Liu, C. Wu, and S. Ji, "Deep graph matching and searching for semantic code retrieval," in *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2021, pp. 1-21.

[10] C. Zeng, Y. Yu, S. Li, X. Xia, Z. Wang, M. Geng, B. Xiao, W. Dong, and X. Liao, "deGraphCS: Embedding Variable-based Flow Graph for Neural Code Search,"*arXiv preprint arXiv:2103.13020*, 2021.

[11] J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The program dependence graph and its use in optimization," in *ACM Transactions on Programming Languages and Systems (TOPLAS)*. ACM, 1987, pp. 319-349.

[12] S. S. Patil, "Automated Vulnerability Detection in Java Source Code using J-CPG and Graph Neural Network," *University of Twente*, 2021.

[13] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated Graph Sequence Neural Networks," in *Proceedings of ICLR'16*, 2016.

[14] F. Lv, H. Zhang, J. -g. Lou, S. Wang, D. Zhang, and J. Zhao, "CodeHow: Effective Code Search Based on API Understanding and Extended Boolean Model (E)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 260-270, doi: 10.1109/ASE.2015.42.

[15] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, "Portfolio: finding relevant functions and their usage," in *Proceedings of the 33rd*

International Conference on Software Engineering. ACM, 2011, pp. 111-120.

[16] J. Shuai, L. Xu, C. Liu, M. Yan, X. Xia, and Y. Lei, "Improving code search with co-attentive representation learning," in *Proceedings of the 28th International Conference on Program Comprehension*. ACM, 2020, pp. 196-207.

[17] S. Fang, Y. S. Tan, T. Zhang, and Y. Liu, "Self-Attention Networks for Code Search," in *Information and Software Technology*, 2021, 134: 106542.

[18] Y. Wan, J. Shu, Y. Sui, G. Xu, Z. Zhao, J. Wu, and P. Yu, "Multi-modal attention network learning for semantic source code retrieval," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 13-25.

[19] S. Liu, X. Xie, J. Siow, L. Ma, G. Meng, and Y. Liu, "GraphSearchNet: Enhancing GNNs via Capturing Global Dependency for Semantic Code Search," *arXiv preprint arXiv:2111.02671*, 2021.

[20] R. Sirres, T. F. Bissyandé, D. Kim, D. Lo, J. Klein, K. Kim, and Y. L. Traon, "Augmenting and structuring user queries to support efficient free-form code search," in *Empirical Software Engineering*, 2018, 23(5), pp. 2622-2654.

[21] L. Xuan , Q. Wang, and Z. Jin, "Description Reinforcement Based Code Search," in *Journal of Software*, 2017.

[22] T. Kenter, A. Borisov, and M. De Rijke. "Siamese cbow: Optimizing word embeddings for sentence representations," *arXiv preprint arXiv:1606.04640*, 2016.

[23] A. Lazaridou, N. T. Pham, and M. Baroni, "Combining language and vision with a multimodal skip-gram model," *arXiv preprint arXiv:1501.02598*, 2015.

[24] P. Lam, E. Bodden, O. Lhoták, and L. Hendren, "The Soot framework for Java program analysis: a retrospective" in *Cetus Users and Compiler Infastructure Workshop (CETUS 2011)*, 2011, 15(35).

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Mapping Modern JVM Language Code to Analysis-friendly Graphs: A Pilot Study with Kotlin

Lu Li
School of Software Engineering
Tongji University
Shanghai, China
2033816@tongji.edu.cn

Yan Liu*
School of Software Engineering
Tongji University
Shanghai, China
yanliu.sse@tongji.edu.cn

***Abstract*—Kotlin is a modern JVM language, gaining adoption rapidly and becoming Android official programming language. With its widely usage, the need for code analysis of Kotlin is increasing. Exposing code semantics explicitly with a properly structured format is the first step in code analysis and the construction of such representation is the foundation for downstream tasks. Recently, graph-based approaches become a promising way for encoding source code semantics. However, current works mainly focus on representation learning with limited interpretability and shallow domain knowledge. The known evolvements of code semantics in new-generation programming languages have been overlooked. How to establish an effective mapping between naturally concise Kotlin source code with graph-based representation needs to be studied by analyzing known language features. In this paper, we propose a first-sight, rule-based mapping method, using composite representation with AST, CFG, DFG, and language features. We evaluate mapping strategies with ablation experiments by simulating a code search solution as a downstream task. Our graph-based method with built-in language features outperforms the text-based way without introducing greater complexity. By addressing the practical barriers to extracting and exposing the hidden semantics from Kotlin source code, our study also helps enlighten source code representations for other modern languages.**

Keywords- Kotlin; graph representation of code; code analysis; language feature;

I. INTRODUCTION

In 2017, Google announced Kotlin as an Android official programming language. It combines object-oriented and functional features. Being a more modern, expressive, and safer programming language, Kotlin has achieved a significant diffusion among developers, the adoption of the Kotlin was rapid. It was the fastest-growing language on GitHub 2018[1] and ranked 7th among mobile development languages in Top Programming Languages 2021 published by IEEE Spectrum[2].

As more and larger codebases written in Kotlin appear, the need for code analysis is increasing. Code analysis is learning from source code through relevant means to solve downstream tasks such as code defect detection, code search, etc., bringing lots of amazing tools and great convenience. The first step of machine learning-based approaches of code analysis is to map source code into intermediate code representation. Generated methods can be categorized as sequence-, tree-, and graph-based[3][4][5], among which graph-based representation has better performance in code analysis these years, becoming a more promising approach for intermediate representation.

However, there is a lack of code analysis related research based on Kotlin for now, and also no related research on the Kotlin graph representation method. Even the graph representation of code itself is still an issue under study, that there is no widely accepted method to convert code into graph representation[6]. Furthermore, Kotlin is a concise language, with less information presented in source code. The usage of language features of Kotlin is also under study. So, no one knows what is a more suitable way to map Kotlin source code to graph representation.

Therefore, in this paper, we conduct a pilot study on how to map Kotlin source code to analytics-friendly graphs. The contribution of this work is as follows:

- This is the first study focused on graph representation of Kotlin code to our knowledge. We proposed a first-sight rule-based feature-enhanced graph mapping method. It can provide other Kotlin downstream task researchers with a basis for constructing graph representation for programs.
- We verify the necessity of studying graph representation of Kotlin separately by comparing the difference between Kotlin and Java in graph representation through the method of induction and summary.
- Through a downstream source-code query task, we proved our graph-based method is more effective than text-based methods. We also proved that language features are useful to enhance graph representation with no greater cost.

II. BACKGROUND

To our best knowledge, there has not been any study on the graph representation method of Kotlin Language. Yet, some Kotlin related studies and graph representation of

* Corresponding author
DOI reference number: 10.18293/SEKE2022-079

program do exist. In this section, we will review the relevant literature.

A. Kotlin

Since Kotlin has 100% interoperability with Java, quite a few Kotlin related studies did a comparative study between Java and Kotlin language[7][8][9]. They concluded that Kotlin is concise and safe, can improve the productivity of the programmer, and also improve the quality of applications.

Besides, there are several empirical studies conducted on its adoption by the developers. Mateus.B.G et al.[10] explored the source code of 387 Android applications, describes the evolution of usage of Kotlin language features on these applications. Martinez.M et al.[11] and Coppola.R et al.[12] did research on the evolution of Java to Kotlin and believed that Kotlin can ensure the seamless migration of Android developers from Java to Kotlin.

The current research on Kotlin mostly stays at analyzing the differences between Kotlin and Java in terms of syntax, performance, language features, as well as the research on the quality and performance of programs written in Kotlin, the study of graph representation of Kotlin is still empty.

B. Graph Representation of source code

There is currently no widely accepted method for mapping programs into graph representation, different studies have different graph representation methods. Allamanis et al.[3] mapped program to graphs consisting of ASTs together with control-flow edges, data-flow edges, and a hand-crafted set of additional typed edges. Lu M et al.[4] proposed a program graph method named FDA, which integrates the AST, function call graph, and data-flow graph to characterize syntax and semantic information. T. T. Nguyen et al.[5] proposed a program graph method named Groum, where nodes represent actions and control points, and edges represent control and data flow dependencies between nodes.

Almost all graph representation methods are coupled to language, like Allamanis et al.'s work[3] is for C#, Lu M et al.'s work[4] is for C++ and Nguyen et al.'s work[5] is for Java. But Kotlin has no related graph representation methods. This is mainly due to the fact that Kotlin is a relatively new language and there is a lack of tools and tagged datasets for further study.

III. ON THE NECESSITY OF STUDYING KOTLIN SEPARATELY

As Java is a mature programming language and Kotlin is so connected with Java, can we directly convert Kotlin to Java, and then map the converted code to graph representation? To verify the necessity of studying Kotlin separately, we compare Kotlin and Java in terms of 3 dimension in this section and found some profound differences that prevent directly using graph representation methods of Java to map Kotlin code.

A. Decompile Kotlin Code to Java Code

Kotlin is 100% interoperable with Java. Kotlin and Java source code can even be converted to each other. Tools in IntelliJ IDEA and Android Studio can help us do such conversion. But there are some problems:

1) Kotlin program must be compiled before decompiled. But it's difficult to successfully compile each projects, for there are always many environment and configuration requirements.

2) There is currently no tool for batch decompilation. We need to decompile Kotlin code file by file if we want to decompile the whole project.

3) The conversion effect is unsatisfactory. The Java file decompiled from Kotlin always has many redundant meaningless encoding and some even have bugs. This also implies that the conciseness of Kotlin makes it lose some information, and it may be harder to analyze kotlin

To sum up, decompiling Kotlin code into Java code is time-consuming and troublesome.

B. Kotlin's Language Features

Kotlin provides programmers with various language features that make it concise, safe and expressive. We collected 30 Kotlin language features as Table I summarized. These features are further summarized on the basis of Mateus B G's work[10]. They extracted 24 Kotlin features from a document that compares Kotlin and Java[13], Kotlin's releases notes, and Kotlin Reference[14]. We inspect these documents and websites, update 6 new features (marked in the table) in our list.

TABLE I. LANGUAGE FEATURES IN KOTLIN

id	feature	id	feature
1	Type inferences	16	singletons
2	Lambda expressions	17	Companion object
3	Inline function	18	Destructing declaration
4	Null-safety	19	Infix function
5	When expressions	20	Tail-recursive function
6	Function w/arguments with a default value	21	Sealed class and sealed interfaces
7	Function w/named arguments	22	Type aliases
8	Smart casts	23	coroutines
9	Data classes	24	contract
10	Range expressions	25	Inline classes
11	Extension functions	26	properties
12	String template	27	Primary constructors
13	First-class delegation	28	Operator overloading
14	Declaration-site variance & Type projections	29	Separate interfaces for read-only and mutable collections
15	Suspending functions	30	Instantiation of annotation classes

We can see that there are plenty of language features that exist in Kotlin but not present in Java. If we convert Kotlin to Java to construct graph representation, we will lose these Kotlin language features in the graph, which contain lots of information and represent Kotlin's characteristics.

C. Verbosity vs Concise

Java is a verbose language, yet Kotlin's syntax focuses on removing verbosity. Rough estimates indicate approximately a 40% cut in the number of LOC compared to Java[8]. Fig. 1 is a Kotlin code snippet and its decompiled Java code. Java code is nearly 3 times longer than Kotlin to implement the same function. Therefore, converting Kotlin to Java to construct graph representation would take away such concise

characteristics in Kotlin, which is one of the most significant features of Kotlin.

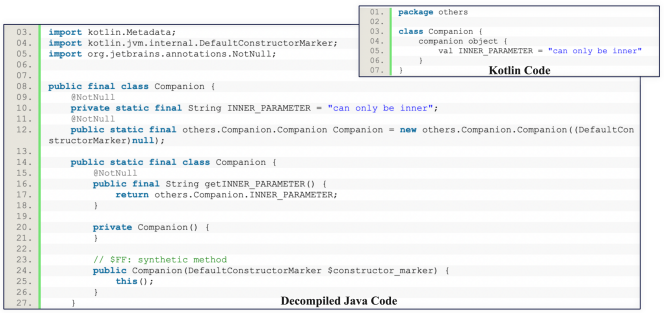


Figure 1. Kotlin code and its decompiled Java code

In conclusion, constructing graph by converting Kotlin to Java is not only troublesome; but also will lose Kotlin's crucial features. Therefore, it is necessary to study the graph representation of Kotlin separately.

IV. GRAPH MAPPING STRATEGY

A. Graph Representation of Code

We use composite code representation and denote Kotlin code by a joint graph with three types of sub-graphs (AST, CFG, and DFG) and enhanced by language features nodes, edge, and attributes. All edges in the graph are directed edges.

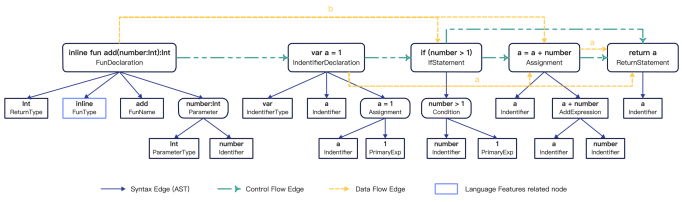


Figure 2. Graph representation example

AST (Abstract Syntax Tree) is the fundamental structure of program and it contains almost all syntax information of program, we use AST as the backbone of the graph representation of Kotlin. The major AST nodes are shown in Fig. 2. All boxes are AST nodes, with specific codes in the first line and node type annotated. The dark blue arrows represent the child-parent AST relations, which are called syntax edges in our methods.

CFG (Control Flow Graph) describes all paths that might be traversed through a program during its execution. The path alternatives are determined by conditional statements, e.g., if, for, and switch statements. In CFGs, nodes are connected by directed edges to indicate the transfer of control. The CFG edges are highlighted with green arrows in Fig. 2. Two different paths derive from the if statement.

DFG (Data Flow Graph) tracks the usage of variables throughout the CFG. Data flow is variable oriented and any data flow involves the access or modification of certain variables. A DFG edge represents the subsequent access or modification onto the same variables. It is shown by yellow arrows in Fig. 2 with the involved variables annotated over the edge. For example, the identifier a is used both in the assignment statement and the return statement.

Language Features extract common patterns in source code, containing much information. We explicitly represent Kotlin language features in the graph, some by adding edges, some by adding nodes, and some by adding attributes to nodes, according to each features' characteristics. In Fig. 2, inline function feature is represented by a node with function type inline that is illustrated by the light blue box.

B. Construction Process

The construction process is shown in Fig. 3, we first use *kotlinx.ast*, a generic AST parsing library, to extract functional external AST. This library only can parse ASTs outside functions, so we need to manually extract ASTs inside functions by traversing code, which is carried out to the statement level. Nodes in ASTs represent statements, and they are classified into 19 types according to their grammatical structure information, such as *IdentifierDeclaration*, *IfStatement*, etc. And we connected these nodes by syntax edges according to their syntactic relationship.

Then, We extract control flow by analyzing ASTs we have already gotten, including syntax information of ASTs and control statements. Then we record the scope of variables and statements that use the variable to get data flow.

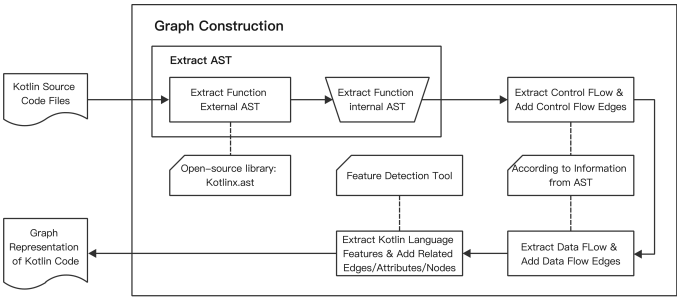


Figure 3. Graph construction process

To extract Kotlin language features, we built a feature detection tool operating on Kotlin source code and ASTs. For each language features presented in Table I, we first manually investigated how a feature is represented in source code. Then we encoded different analyzers for detecting feature instances on source code files. For features that cannot be detected in source code, we extracted them by analyzing raw ASTs. Then, we add these features information into the joint graph.

C. Points to Note

Some points need to note in the graph construction process, including problems, limitations, and some hints.

Lack of tools Kotlin is a relatively new programming language and there is a lack of tools, which creates difficulties for Kotlin code analysis. For example, there are only two AST parsing libraries for AST, *kotlinx.ast*[2] and *kastree*[3]. But both of them are with limited function and

[2] "kotlinx/ast", https://github.com/kotlinx/ast, 2022-03-11.
[3] "kastree", https://github.com/cretz/kastree, 2022-03-11.

cannot satisfy our requirements. Therefore, if one needs to generate Kotlin graph representation in large batches, you should develop a robust Kotlin AST parser first.

Maintain a Kotlin language feature diagram Kotlin is an actively released language, new language features will be introduced in future releases. In order to ensure the integrity of the language features that can be represented in the graph, one should maintain a Kotlin language feature form. Every time there is a new release, one should check the release note and keep the diagram up to date.

Feature Representation Method Different language features have different representation methods including adding edge, adding nodes, and adding attributes to nodes, according to their characteristics. Each features need to be investigated separately to decide how to represent it in the graph is more reasonable and analysis-friendly.

V. EXPERIMENT

In this section, we describe our experiment and discuss the result. Our experiment is guided by the following research questions:

- RQ1: How our graph-based method performs comparing to text-based methods?
- RQ2: Whether is it effective to add language features to graph representation?

A. Experiment Framework for Down-stream Task

Software developers and tools often query source code to explore a system, or to search for code for maintenance tasks such as bug fixing, refactoring, and optimization. Considering the objectivity of the downstream task, we choose Wiggle[15], a representative source code query system based on the graph data model as a reference for our downstream tasks. We transform it to support Kotlin query, forming a code query framework for our experiment as shown in Fig. 4.

We develop a graph representation constructor as we describe in Section IV, mapping Kotlin source code to graph representation. Then we store graph representation into a Neo4j graph database. For a given query, the framework would return the code excerpts found (labeled with their location).

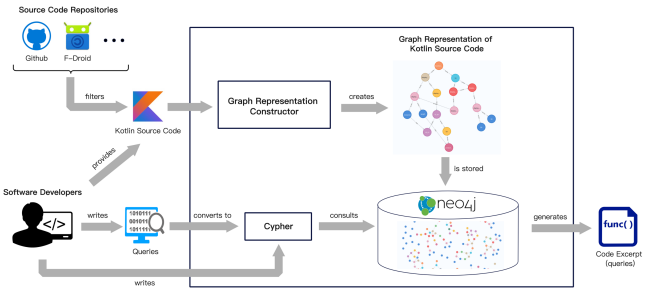


Figure 4. Source code query framework

B. Experiment Setup

1) Basic Setup

We executed various queries for different scenarios on a corpus of three Kotlin Projects on GitHub (shown in Table II). The dataset is relatively small because of the lack of tools as we noted in Section IV, limiting us generate graph representation in a large batch. But the data volume of this size is sufficient for our study. After converting all source code into graph representation and storing them into Neo4j, there are 2462 nodes and 2556 edges (relationships) in the database.

TABLE II. DATASET

Dataset	Description	LoC
Kotlin101	A collection of runnable console applications that highlights the features of Kotlin.	747
KAndroid	Kotlin library for Android to eliminate boilerplate code in Android SDK and focus on productivity.	886
Android-SearchView	A demonstration application for android's SeachView.	415

All the tests commence on a MacBook Pro with 8-core CPU, 16GB unified memory, and 512GB SSD. The graph database Neo4j Browser version is 4.4.2 and Server version is 4.3.10 (community).

2) Query Selection

We select 11 queries in three query scenarios, which consist of language research, complex search, and program check. Language research and complex search refer to Wiggle's query examples[15] and program check refer to the evaluation part in Rodriguez-Prieto O et al.'s work[16]. We modified queries in their work, forming 11 queries shown in Table III.

TABLE III. SAMPLE QUERIES

id	Purpose	Query
Q1	Language Research	lambda expression
Q2		companion object
Q3		for statement
Q4		Inline function
Q5	Complex Search	function with *Int* return type
Q6		search for classes containing recursive methods
Q7		find instances of classes that inherited from *People*
Q8	Program Check	Binary conditions prefer *if* over *when*
Q9		Public functions/methods that return platform type expressions must explicitly declare their Kotlin type
Q10		Return an empty array or collection instead of a null value for methods that return an array or collection
Q11		Convert integers to floating point for floating-point operations

C. Result and Discussion

- **RQ1: How the graph-based method performs compare to text-based methods?**

We provide a comparative study of text-based methods and our graph-based method to answer this question. We choose keyword match and regular expressions as text-based source code search approach in our comparative study, which is the most common and widely used text-based approaches for searching code for now.

First, we evaluate the coverage of these three approaches. The result is shown in Table IV. Obviously, graph-based method can cover more queries than the other two text-based approaches, especially more complex search including more dependency and requirements. This is mainly because that code is texts with structures and semantics, such information is implicit that these text-based search approaches are unable to capture. In contrast, graph representation of code contains plentiful syntax and semantic information, provided by nodes and their relations, which can be leveraged for effective search.

TABLE IV. COVERAGE OF DIFFERENT APPROACHES

Query_id	keyword	Regular expression	Graph representation
Q1		√	√
Q2	√	√	√
Q3	√	√	√
Q4	√	√	√
Q5	√	√	√
Q6			√
Q7			√
Q8		√	√
Q9			√
Q10	√	√	√
Q11			√

Then, for queries that text-based methods and graph-based method all can cover, we conduct further evaluation by introducing two performance indicators. HitRate is the percentage of correct search results out of all correct results, evaluating the exhaustion of search approaches. P@all evaluate the precision of search results. It calculates by the percentage of correct search results out of all search results. The result is shown in Table V.

TABLE V. TEST RESULT

id	HitRate			P@all		
	Key word	*Regular expression*	*Graph-based*	*Key word*	*Regular expression*	*Graph-based*
Kotlin101						
Q1	N/A	20%	100%	N/A	60%	100%
Q2	100%	100%	100%	70%	50%	100%
Q3	100%	100%	100%	67%	100%	100%
Q4	100%	100%	100%	70%	50%	100%
Q5	100%	100%	100%	17%	100%	100%
Q8	N/A	100%	100%	N/A	100%	100%
Q10	100%	100%	100%	25%	67%	100%
KAndroid						
Q1	N/A	26%	100%	N/A	100%	100%
Q2	-	-	-	-	-	-
Q3	100%	100%	100%	4%	100%	100%
Q4	100%	100%	100%	93%	100%	100%
Q5	100%	100%	100%	10%	80%	100%
Q8	N/A	100%	100%	N/A	100%	100%
Q10	0%	100%	100%	0%	20%	100%
Android-SearchView-Demo						
Q1	N/A	0%	100%	N/A	0%	100%
Q2	100%	100%	100%	100%	100%	100%
Q3	100%	100%	100%	40%	67%	100%
Q4	-	-	-	-	-	-
Q5	100%	100%	100%	91%	100%	100%
Q8	N/A	100%	100%	N/A	100%	100%
Q10	100%	100%	100%	33%	50%	100%

For both HitRate and P@all, Graph-based approach has significantly better performance, especially for P@all. Text-based approaches has low P@all, because they often return some code that matches the query but is not related, like description in comment or unrelated code contains keyword or match the regular expression. Fig. 5 shows some unrelated results examples with their corresponding queries in text-based methods for Q5. Better performance of graph-based methods shows that preferentially extracting language features and add them to graph representation is more promising than extracting them directly from text.

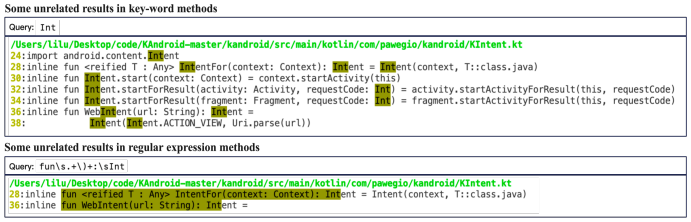


Figure 5. Unrelated results examples in text-based methods

- **RQ2: Whether is it effective to add language features to graph representation?**

To assess the coverage of our approach, we analyze which code representations are necessary to describe different kinds of queries. The results of this analysis are presented in Table VI.

Obviously, the AST alone provides only a little information for querying source code. By combining AST with CFG or DFG, we obtain a better view of the code and can describe almost every query except language feature-related query. But language features contain much information and represent characteristics of Kotlin, so language features related queries should not be excluded. After adding language features, we are finally able to model all the query samples, making use of information available from AST, CFG, DFG, and language features representation.

TABLE VI. COVERAGE OF DIFFERENT GRAPH REPRESENTATION

id	AST	AST+ CFG	AST+ DFG	AST+language features	AST+CFG+ DFG+language features
Q1				√	√
Q2				√	√
Q3	√	√	√	√	√
Q4				√	√
Q5	√	√	√	√	√
Q6		√			√
Q7		√			√
Q8	√	√	√	√	√
Q9	√	√	√	√	√
Q10	√	√	√	√	√
Q11			√		√

We also calculated the graph complexity of different graph representations as shown in Table VII. The graph complexity measurement method uses the graph measures proposed by Dehmer M et al[17] using a polynomial-based approach. Through the comparison, we can see that adding language

features in the graph representation will not increase the complexity greatly, but it really can represent more information.

TABLE VII. COMPLEXITY OF DIFFERENT REPRESENTATION

Dataset	AST	AST+CFG+DFG	AST+CFG+DFG+ language features
Kotlin101	0.28351	0.35987	0.37066
KAndroid	0.16175	0.20886	0.21384
SearchView	0.36623	0.41296	0.42089
Average	0.27049	0.32723	0.33513

VI. THREAT TO VALIDITY

ASTs inside functions. Because of the lack of tools, we need to manually extract ASTs insides functions by using types of nodes and edges defined by ourselves. So ASTs inside functions may lack of unity. When nodes and edges are defined differently, the results are different.

Feature representation method. Different language features have different representation methods including adding edges, adding nodes, and adding attributes to nodes, according to their characteristics. Feature representation in this paper is all reasonable, but anyway, there will be better ways of representation, which also will affect the results.

Queries of text-based methods. Though key-word method and regular expression method is standardized, queries is not. Different queries will lead to different results.

VII. CONCLUSION

In this paper, we conduct a pilot study on graph representation method of Kotlin source code. We first verify the necessity of studying graph representation of Kotlin separately by comparing the difference between Kotlin and Java in graph representation through the method of induction and summary. Then we proposed a first-sight, rule-based graph mapping method. It takes AST as the skeleton, enriching with control flow edge and data flow edge, together with some edges and attributes representing Kotlin's language features ostensive. We present our graph construction process and summarized points to note in the process, aiming to provide other Kotlin downstream task researchers with a basis for constructing graph representation for programs.

We evaluate our method through a source-code query down-stream task, came to the following conclusions: 1) Graph representation methods outperform text-based methods both on query coverage and search result. This is because graph representation contains more syntax and semantic information which can be well leveraged in source-code search and even other source-code analysis tasks. 2) Language features are useful to enhance graph representation. First, graph representation with language features can cover more queries. In addition, preferentially extracting language features and adding them to graph representation is more promising than extracting them directly from text. 3) Adding language features to graph representation will not add much complexity.

In the future, we plan to study next steps of code analysis for Kotlin, including graph embedding, neural network, and so on, aiming to conduct a full-link study on code analysis for Kotlin and promote its application in the field of “big code”. Furthermore, our study with Kotlin is instructive for the analysis of similar concise modern languages, that adding language features to graph representation is an exploration direction.

REFERENCES

[1] “The State of the Octoverse 2018,” https://octoverse.github.com/2018/projects#languages, last access: 2022-03-11.

[2] “Top Programming Languages 2021,” https://spectrum.ieee.org/top-programming-languages/, last access: 2022-03-11.

[3] Allamanis M, Brockschmidt M, Khademi M. Learning to represent programs with graphs[J]. arXiv preprint arXiv:1711.00740, 2017.

[4] Lu M , Tan D , N Xiong, et al. Program Classification Using Gated Graph Attention Neural Network for Online Programming Service[J]. 2019.

[5] T. T. Nguyen, H. A. Nguyen, N. H. Pham, J. M. AI-Kofahi and T. N. Nguyen, "Graph-based mining of multiple object usage patterns", Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering ser. ESEC/FSE '09, pp. 383-392, 2009.

[6] Allamanis M. Graph Neural Networks in Program Analysis[M]//Graph Neural Networks: Foundations, Frontiers, and Applications. Springer, Singapore, 2022: 483-49

[7] Gotseva D, Tomov Y, Danov P. Comparative study Java vs Kotlin[C]//2021 27th National Conference with International Participation (TELECOM). IEEE, 2021: 86-89.

[8] Flauzino M, Veríssimo J, Terra R, et al.. Are you still smelling it? A comparative study between Java and Kotlin language[C]//Proceedings of the VII Brazilian symposium on software components, architectures, and reuse. 2018: 23-32.

[9] Mateus B G, Martinez M. An empirical study on quality of Android applications written in Kotlin language[J]. Empirical Software Engineering, 2019, 24(6): 3356-3393.

[10] Mateus B G, Martinez M. On the adoption, usage and evolution of Kotlin features in Android development[C]//Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2020: 1-12.

[11] Martinez M, Mateus B G. How and Why did developers migrate Android Applications from Java to Kotlin? A study based on code analysis and interviews with developers[J]. arXiv preprint arXiv:2003.12730, 2021.

[12] Coppola R, Ardito L, Torchiano M. Characterizing the transition to Kotlin of Android apps: a study on F-Droid, Play Store, and GitHub[C]//Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics. 2021: 8-14.

[13] 2016. Comparison to Java Programming Language. https://Kotlinlang.org/docs/ reference/comparison-to-Java.html Online; accessed 01-July-2019.

[14] JetBrains. 2019. Kotlin Language Documentation. https://Kotlinlang.org/docs/Kotlin-docs.pdf.

[15] Urma R G, Mycroft A. Source-code queries with graph databases—with application to programming language usage and evolution[J]. Science of Computer Programming, 2015, 97: 127-134.

[16] Rodriguez-Prieto O, Mycroft A, Ortin F. An efficient and scalable platform for Java source code analysis using overlaid graph representations[J]. IEEE Access, 2020, 8: 72239-72260.

[17] Dehmer M, Chen Z, Emmert-Streib F, et al. Measuring the complexity of directed graphs: A polynomial-based approach[J]. Plos one, 2019, 14(11): e0223

Refactoring of Object-oriented Package Structure Based on Complex Network

Youfei Huang, Yuhang Chen, Zhengting Tang, Liangyu Chen, Ningkang Jiang*,
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University,
Shanghai, China,
nkjiang@sei.ecnu.edu.cn

***Abstract*—A software system is usually developed with multiple modules. However, its structure is continuously modified during software evolution, resulting in poor maintainability and understandability. Therefore, software evolution must accompany system refactoring. This paper describes an optimization approach for package structure according to complex network theory. First, we analyze the relations between classes and build the class dependency graph. Second, we propose a community detection algorithm to recombine the classes and optimize system cohesion and coupling without changing the external functionality. Third, by comparing the original and optimized package structure, the two dimensions of splitting the package and moving classes between packages identify package refactoring opportunities. In addition, we evaluate the impact of the above approach on package quality in terms of package reusability and instability. We design experiments on 10 open-source Java software projects to verify the effectiveness of our approach.**

***Index Terms*—Refactoring;complex network;community detection;cohesion;coupling**

I. Introduction

During the software life cycle, new requirements will emerge inevitably, and software modifications to implement new requirements may not conform to object-oriented programming specifications [1]. In addition, the tight development cycle may lead to poor design decisions [2]. The differences between the system and the original design increase in the long run, and the quality of the system becomes increasingly poor. It becomes more and more challenging to maintain the existing software. In order to avoid the fate of software system corruption, fragmentation and even disintegration, choosing a proper refactoring operation is a feasible method. Refactoring can adjust the software architecture without changing the code's external behavior, optimize the existing code, and extend the software system's life [3].

In the process of software development and maintenance, the problem of Too Large Packages (TLP) or Too Small Packages (TSP) may occur in the system. When a package contains more than 30 classes or exceeds 27,000 code lines [4], it can be considered as TLP. The number of classes or lines of code in a package that the TSP should contain is not explicitly given in [4]. According to previous research, when the number of classes in a package is 4 or less [5, 6], it is usually not worth the effort to maintain them.

DOI reference number: 10.18293/SEKE2022-164

Complex network theory has been applied to many fields, such as social networks [7, 8], computer networks [9], and biological networks [10]. One of the notable features of complex networks is their community structure [11, 12]. Software network built with classes as nodes and dependencies between classes as edges is the product of characterizing software systems as complex networks, so it has community structure property [13]. Based on complex network theory, we propose a system-level automatic package refactoring approach that abandons the original package structure of the system and re-modularizes the system. First, we represent the software system as a class dependency graph. Second, based on the definition of TLP and TSP, we propose a community detection algorithm to detect the community structure corresponding to the optimized package structure. Third, by comparing the optimized package structure with the original package structure, we give refactoring suggestions. Through tests on 10 open source software systems, it is verified that our refactoring approach is meaningful and positively impacts the cohesion of the system while reduces the coupling of the system. The primary contributions of this study are summarized as follows.

- From the perspective of optimizing system structure, a controllable community detection algorithm that divides the size of sub-communities is proposed, which can obtain the best class distribution from the system-level, and eliminate too large and small packages in the system.
- Several open-source software systems are used to evaluate our approach. The evaluation metrics based on cohesion and coupling metrics are compared with previous studies, and we also assess the changes in system reusability and stability after refactoring.

The remainder of the paper is organized as follows. Section II summarizes the related work. Section III presents the package refactoring algorithm. In Section IV, we design experiments to verify the effectiveness of our approach. Section V is the conclusion.

II. Related Work

A. Package-level Refactoring

Over the last three decades, software engineers have proposed several semi-automatic and full-automatic refactoring methods to improve software quality. Depending on the kind

of entities selected in refactoring, there are three main types of refactoring at different granularities: package level, class level, method and property level. In this paper, we focus on package level refactoring. Pan et al. [14] represented the package, class and their dependencies with weighted bipartite software networks and proposed a guidance community detection algorithm to optimize the package structure of the software system. Mi et al. [15] divided the dependency relationships between classes into five types to build a class dependency graph, and then proposed a cohesion metric at the package level, according to this metric to move class between packages. Zhou et al. [16] applied Mi's approach to build software networks, proposed a coupling metric of packages and improved the structure of packages considering changes in cohesion and coupling values. Bavota et al. [5] combined semantic and structural metrics to generate a class-by-class matrix, where the values in the matrix indicate the likelihood of two classes being in the same package, after which the strongly related class chains in the matrix are extracted, and the classes in one class chain are placed in the same package. However, their approach only focuses on refactoring one package at a time, and incrementally re-modularizing a software system. Abdeen et al. [17] and Chhabra et al. [18] used the multi-objective Non-Dominated Sorting Genetic Algorithm to optimize system structure by moving classes between packages, while respecting the original package organization as much as possible, to increase cohesion and reduce coupling and cyclic connectivity.

B. Evaluation Metrics of Package Quality

Since the software is frequently changing, software designers should assess software quality periodically. The evaluation standard is the authoritative software metrics proposed by research and engineering. It plays an important role in many fields in the life-cycle of the software, including predicting software defects and maintenance costs [19], and if they are appropriately selected and applied, improvements can be identified and quantified. Wang et al. [2] used the Quality Model for Object-Oriented Design (QMOOD) metric proposed by Harrison et al. [20] to evaluate the improvement in reusability, flexibility, and understandability of the system after their refactoring. QMOOD is a quality metric at the class level in object-oriented design, and Singh et al. [21] proposed the Quality Metric of Package Level in Object-Oriented Design (QMPOOD) with quality attributes including reusability, flexibility, functionality, understandability, extendibility, and effectiveness, and later they gave a specific method to calculate the reusability of software system packages in [22]. Chong et al. [23] presented a weighted complex network to represent the structural characteristics of object-oriented software systems and used 40 object-oriented software systems for experiments to evaluate the maintainability and reliability of software systems. Martin et al. [24] proposed software package metrics based on object-oriented design principles, including eight metrics: efferent coupling (C_e), afferent coupling (C_a), instability (I), number of abstract classes (Na), number of classes (N_c), abstractness (A), the distance from the main sequence (D) and the normalized distance from the main sequence (D_n).

III. METHODOLOGY

A. Problem Formulation

We use the code snippet shown in Fig.1 as a motivation example to formulate our problem. In Fig.1, there are 11 classes, which are divided into 4 packages. Classes do not exist independently, but some classes are more dependent on classes in other packages. Therefore, there are "bad smells" in the code.

P1

```
class A{
   A a = new A();
}
classB{
   public A objA;
   public B objB;
   public void m1(A a){}
}
```

P2

```
import p1.B;
class C extends B implements D{
   public B m2(){
      B b = new B();
      return b;
   }
   public void m3(){}
}
interface D{
   public void m3();
}
class E{
   public void m4(){}
}
```

P3

```
import p2.E;
class F{
   public void m5(){
      E e = new E();
      e.m4();
   }
}
class G extends F{
   public F objF;
   public void m6() throws H{}
}
class H extends Exception{}
```

P4

```
import p3.F;
class I extends F{
   public F objF;
   public void m7(F ff){
   }
}
class J{
   public K m8(){
      K k= new K();
      return k;
   }
}
class K {
   public I objI;
}
```

Fig. 1. Code snippet used as an example to understand the proposed algorithm.

Based on five class dependencies outlined in [15], which are inheritance and implementation, aggregation, parameter, signature, and invocation. We extract the associations between classes in Fig.1 and then build the Class Dependency Graph (CDG), shown in Fig.2.

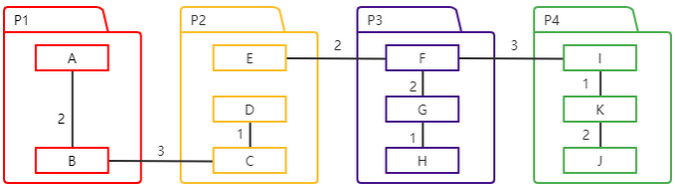


Fig. 2. Class dependency graph

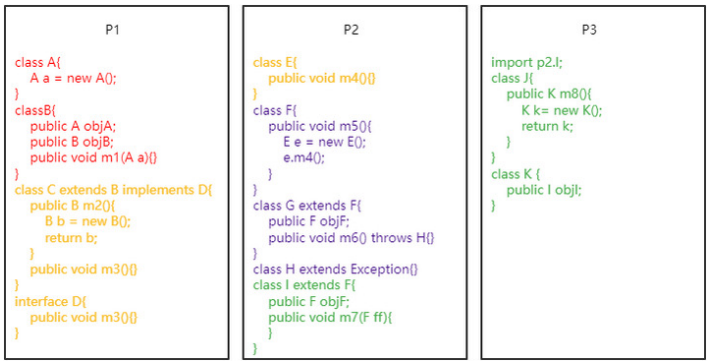


Fig. 3. Optimized package structure.

After refactoring, we get three new packages, shown in Fig.3. The optimized package structure gains a better modularity Q, increased from 0.2699 to 0.5450.

For our refactoring approach, this paper mainly studies the following research questions:

- RQ_1: Can our approach alleviate the design problems and get meaningful refactoring?
- RQ_2: Does our approach have more advantages compared with other refactoring approaches?
- RQ_3: Besides cohesion and coupling, does our approach improve other package design metrics?

B. Method Overview

Our refactoring approach is shown in Fig.4. First, we model the package topology of an object-oriented software system as a weighted software network, with classes as vertices and dependencies between classes as the edges of the network. Second, based on complex network theory, this paper uses a community detection algorithm to recombine the classes and find the communities corresponding to the optimized packages. Third, by comparing the optimized package structure with the original package structure, we obtain the package refactoring opportunities.

C. Package Refactoring Algorithm

With Java software systems as research objects, we analyze the bytecode files of the software system and regard classes and their dependencies as entities. In order to improve the package structure, the refactoring approach in this paper regards each class in the system as an independent community, and the classes in the software system are gradually reaggregated to form several packages by using the community detection algorithm. The number of sub-communities of the community detection algorithm, that is, the total number of packages after system optimization, is not specified in advance, so the number of packages may be different from the original system.

Community detection, also known as community discovery, is a technique used to reveal network aggregation behavior. The nodes in the same community are densely connected, and the connections between nodes in different communities are sparse. Newman et al. [25] proposed to calculate the modularity of unweighted undirected networks. Modularity is a property of a network and a measure of the quality of a particular division of a network. However, this paper builds a weighted software network. Therefore, we appropriately modify the calculation method of modularity and propose a weighted modularity Q, which is defined as

$$\begin{aligned} Q &= \frac{1}{2W}\sum_{ij}\left(w_{ij}-\frac{h_i h_j}{2W}\right)\delta\left(c_i,c_j\right) \\ &= \sum_{p=1}^{n}\left[\frac{W_p}{W}-\left(\frac{H_p}{2W}\right)^2\right], \end{aligned} \tag{1}$$

where W is the sum of all the edge weights, w_{ij} is the weight of the edge between node i and node j, h_i is the sum of the weights of all edges connected to node i, c_i and c_j are the communities to which nodes i and j belong, respectively. If i and j are in the same community, then $\delta\left(c_i,c_j\right)=1$, otherwise $\delta\left(c_i,c_j\right)=0$. And n is the number of communities in the network, W_p is the sum of the weights of the edges within community c_p, H_p is the sum of the weights of all edges connected to community c_p. Based on (1), when merging communities, the modularity change of the system can be calculated by (2), $H_{in}(c_i,c_j)$ represents the sum of all the edge weights in the sub-community formed by the merging of community c_i and c_j.

$$\Delta Q_{ij}=\begin{cases}\frac{H_{in}(c_i,c_j)-H_iH_j}{2W} & \text{if } c_i \text{ connects with } c_j\\ 0 & \text{otherwise,}\end{cases} \tag{2}$$

We propose a community detection algorithm, which takes into account the goal of this package refactoring and avoids too large or small packages. In order to prevent excessive software refactoring, when a community and multiple target communities have the same modularity change after merging, we prioritize merging the community pair with entities that are all defined in the same original package to maintain the original design as much as possible. The refactoring algorithm is shown in Algorithm 1:

Algorithm 1 Package refactoring algorithm

Input: The adjacency matrix $M_{n\times n}$ of CDG.
Output: Community set C; Weighted modularity Q.

1. Let the weighted modularity $Q=0$. The number of initial communities is the number of classes in the system.
2. We calculate the change of the Q of the system after the sub-community is merged according to (2), and calculate the modularity increment matrix ΔQ.
3. we find the maximum element ΔQ_{ij} in ΔQ, and merge the two communities C_i and C_j. And then, we recalculate the modularity increment matrix according to Step 2. Communities are gradually aggregated until the maximum element $\Delta Q_{ij}\leq 0$. Then Q reaches the optimum. We obtain the community set C_1, and the weighted modularity value $Q=Q_1$.
4. According to the definition of TLP, we split the large communities existing in the C_1 as follows: for the large

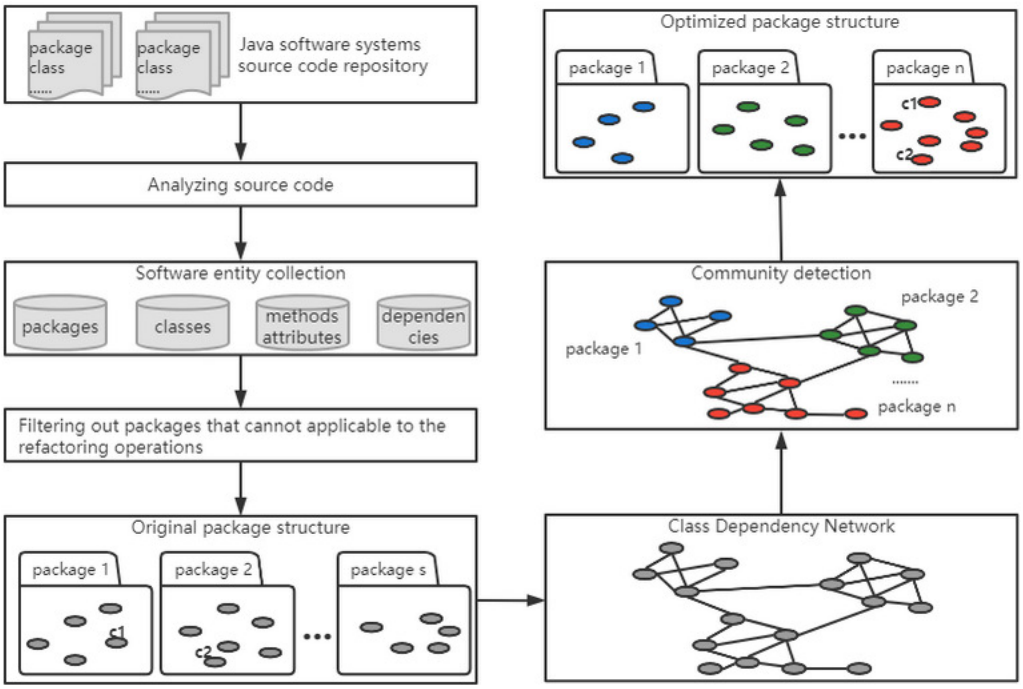


Fig. 4. The workflow of the refactoring approach.

community, we delete the edges with smaller weight in turn. In a splitting round, the edge with the smallest weight is chosen, and the community is split into several connected components, where each component has no more than 30 nodes. If not, split recursively. Finally, we obtain the community set C_2, and the weighted modularity value $Q = Q_2$. 5. For TSP in C_2, we relocate them to other communities with the following merging method: according to the reduction value of modularity value Q, we merge the small sub-community with the target community with the smallest reduction of Q, at the same time, the number of nodes in the newly generated community cannot exceed 30, otherwise we merge with the target community with the second smallest reduction of Q.

IV. Experiments

A. Data Sets

The experiments are conducted on a computer with i5-3230M CPU, 8G DDR3 Memory, Windows 10. We select 10 open-source Java software systems to verify the effectiveness of the refactoring approach. Our choice of software systems is not random, as they are projects in different application fields. In the future, we will use more systems to verify the effectiveness of our approach. To remove the unrelated files, we filter the experimental data from the following three aspects:

- Only the classes in top-level packages are considered for experiments.
- Utility modules do not participate in the refactoring process.

TABLE I
Detailed information about 10 Java software systems

System	System	PN	CN	EN
$S0$	Cglib-nodep 3.2.6	9	198	961
$S1$	Codec 1.15	7	139	278
$S2$	Emma 2.0.5312	10	140	506
$S3$	Empire-db 2.5.0	21	178	1097
$S4$	GistoolkitSource 2.8.1	64	504	2228
$S5$	ITtracker 3.1.5	38	422	1803
$S6$	Rng 1.3	19	346	729
$S7$	Roller 5.1.1	61	541	2707
$S8$	Tomcat 9.0.1	42	619	1589
$S9$	XMLgraphics-commons-2.6	35	363	801

- Third-party libraries are excluded since they are not parts of software systems.

Table I summarizes the information of 10 systems after preprocessing, including name and version number, number of package(PN), number of class(CN), number of edges in the software network (EN).

B. Changes of the package structure

We follow the steps in Fig.4 to perform the refactoring operation. Table II shows the change of package structure before and after refactoring. Note that PN represents the number of packages, TLP and TSP represent the threshold of too large and small package, respectively, It is observed that our approach can solve the problem of too small packages and too large packages in the software system.

C. Evaluations of Cohesion and Coupling Metrics

In our experiments, we use *COHM* metric [15] to measure the cohesion of packages and *COUM* metric [16] to evaluate

TABLE II
DETAILED INFORMATION ABOUT PACKAGE STRUCTURE

System	Before refactoring			After refactoring		
	PN	TLP	TSP	PN	TLP	TSP
$S0$	9	1	2	8	0	0
$S1$	7	1	1	7	0	0
$S2$	10	1	4	5	0	0
$S3$	21	1	9	13	0	0
$S4$	64	2	27	36	0	0
$S5$	38	2	10	25	0	0
$S6$	19	5	5	19	0	0
$S7$	61	3	20	32	0	0
$S8$	42	6	9	35	0	0
$S9$	35	1	8	28	0	0

the coupling between packages. A higher value of *COHM* indicates a better cohesion of the package. And the lower value of *COUM* indicates the better coupling of the package. We apply these two metrics to the software systems to measure the improvement by refactoring. Table III records the detailed change of *COHM* and *COUM*. We can see the cohesion are increased and the couping are decreased on all systems, which means the structures of all systems are optimized and the refactorings are useful. Thus, we answer the question RQ_1.

TABLE III
CHANGES IN COHESION AND COUPLING METRIC VALUES

System	COHM	COUM
	Before/After/Diff.	Before/After/Diff.
$S0$	0.273/0.365/+0.092	0.163/0.109/-0.054
$S1$	0.409/0.636/+0.227	0.192/0.072/-0.120
$S2$	0.360/0.516/+0.156	0.236/0.161/-0.075
$S3$	0.130/0.226/+0.096	0.404/0.370/-0.034
$S4$	0.148/0.449/+0.301	0.516/0.222/-0.294
$S5$	0.099/0.284/+0.185	0.685/0.445/-0.240
$S6$	0.328/0.504/+0.176	0.412/0.203/-0.209
$S7$	0.174/0.301/+0.127	0.495/0.227/-0.268
$S8$	0.415/0.579/+0.164	0.273/0.191/-0.082
$S9$	0.372/0.634/+0.262	0.294/0.114/-0.180

D. Comparison with Previous Research

We have re-implemented Zhou's [16] work and Abdeen's [17] work on package refactoring. Zhou's approach optimizes the package structure by considering both cohesion and coupling measures, moving the class between packages, and finally selecting the package with better cohesion and coupling as the target package for refactoring the current class. Abdeen's approach is a more conservative optimization of the package structure, using the Non-Dominated Sorting Genetic Algorithm to refactor the package. We compare our refactoring approach with theirs, and Table IV presents the specific changes of cohesion and coupling after refactoring with using three different approaches. The optimal values are presented in bold. It is clear that the cohesion and coupling obtained by our method outperform consistently the other two methods. Therefore, we answer the question RQ_2.

TABLE IV
COMPARISON OF THREE REFACTORING APPROACHES

System	COHMaf	COUMaf
	Our/Zhou/Abdeen	Our/Zhou/Abdeen
$S0$	**0.365**/0.331/0.333	**0.109**/0.159/0.135
$S1$	**0.636**/0.562/0.543	**0.072**/0.080/0.089
$S2$	**0.516**/0.406/0.439	**0.161**/0.174/0.176
$S3$	**0.226**/0.181/0.204	**0.370**/0.371/**0.370**
$S4$	**0.449**/0.249/0.289	**0.222**/0.276/0.337
$S5$	**0.284**/0.140/0.196	**0.445**/0.492/0.523
$S6$	**0.504**/0.372/0.364	**0.203**/0.301/0.305
$S7$	**0.301**/0.223/0.245	**0.227**/0.344/0.366
$S8$	**0.579**/0.449/0.493	**0.191**/0.222/0.229
$S9$	**0.634**/0.437/0.417	**0.114**/0.216/0.199

E. Evaluations of Reusability and Instability Metrics

In this section, we investigate whether our refactoring approach optimizes the design quality of the package, besides cohesion and coupling metric. We focus on reusability and instability. Reusability [22] reflects the ability of a design to be reused in multiple contexts. The higher its value, the higher the reusability of the package. Martin proposed instability in [24] to describe the system stability, the lower its value, the more stable of the package.

Table V records the changes in reusability and instability after refactoring. One can observe that the reusability values are increased while the instability values are decreased. This means the design of all software systems is improved after refactoring. We also visualize the change of reusability and instability in Fig.5 and Fig.6, respectively, for a better understanding the change. Therefore, our approach not only improves the cohesion and coupling metrics, but also improves the design quality of system packages in terms of reusability and instability. So we answer the question RQ_3.

TABLE V
CHANGES IN REUSABILITY AND INSTABILITY METRIC VALUES

System	Reusability	Instability
	Before/After/Diff	Before/After/Diff
$S0$	16.071/18.115/+2.044	0.710/0.549/-0.161
$S1$	17.182/17.290/+0.108	0.671/0.557/-0.114
$S2$	9.490/12.189/+2.699	0.519/0.489/-0.030
$S3$	7.312/14.230/+6.918	0.625/0.576/-0.049
$S4$	6.952/17.468/+10.516	0.515/0.510/-0.005
$S5$	9.920/15.941/+6.021	0.693/0.497/-0.196
$S6$	13.604/13.733/+0.129	0.676/0.594/-0.082
$S7$	8.072/15.601/+7.529	0.599/0.533/-0.066
$S8$	11.444/13.826/+2.382	0.530/0.484/-0.046
$S9$	8.927/11.269/+2.342	0.515/0.386/-0.129

F. Threats to Validity

The internal validity threat to our study is that the weights of five dependencies between classes are equal in building class dependency network, whereas the priorities of the five dependencies should be different according to other classical theories of software. But to the best of our knowledge, no research has given the specific weight assignments that these five dependencies apply to various systems or even a

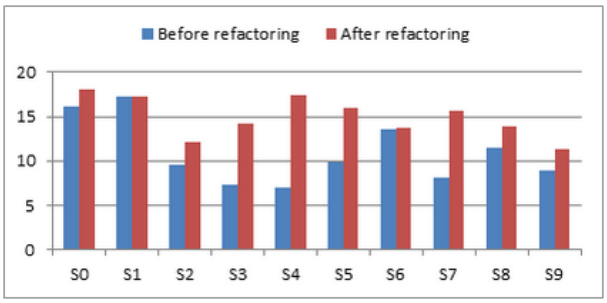


Fig. 5. Reusability change on 10 systems after refactoring.

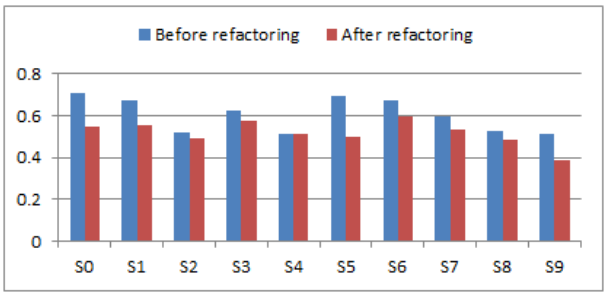


Fig. 6. Instability change on 10 systems after refactoring.

rough range. Considering the importance of five dependencies between classes as the same to build a software weighted network, the effectiveness has been proved in [14], so this part of the threat can be mitigated to a certain extent.

The external validity threat to our study is the limitation of software systems chosen in the experiments. In this paper, we mainly use Java software systems, but there are object-oriented software systems developed in other programming languages such as C++, Python, etc. Therefore, applying our research to projects developed in other programming languages may lack the ability to give accurate refactoring recommendations.

V. Conclusion

In this study, we propose a refactoring approach for package structure based on complex network theory. It uses a community detection algorithm to find opportunities for package refactoring, which achieves the optimal class distribution and get no too large or small packages. The paper analyzes five kinds of dependencies between object-oriented software system classes, which are used to build class dependency network, and then perform refactoring operations according to the software design principle of "high cohesion and low coupling". Experimental results demonstrate that our approach can solve the problem of system cohesion and coupling while maintain the external functionality, and improve software stability and reusability.

References

[1] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Transactions on software engineering*, vol. 30, no. 2, pp. 126–139, 2004.

[2] Y. Wang, H. Yu, Z. Zhu, W. Zhang, and Y. Zhao, "Automatic software refactoring via weighted clustering in method-level networks," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 202–236, 2017.

[3] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[4] M. Lippert and S. Roock, *Refactoring in large software projects: performing complex restructurings successfully*. John Wiley & Sons, 2006.

[5] G. Bavota, A. De Lucia, A. Marcus, and R. Oliveto, "Using structural and semantic measures to improve software modularization," *Empirical Software Engineering*, vol. 18, no. 5, pp. 901–932, 2013.

[6] R. A. Bittencourt and D. D. S. Guerrero, "Comparison of graph clustering algorithms for recovering software architecture module views," in *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE, 2009, pp. 251–254.

[7] J. M. Hofman, A. Sharma, and D. J. Watts, "Prediction and explanation in social systems," *Science*, vol. 355, no. 6324, pp. 486–488, 2017.

[8] H. Ebel, L.-I. Mielsch, and S. Bornholdt, "Scale-free topology of e-mail networks," *Physical review E*, vol. 66, no. 3, p. 035103, 2002.

[9] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[10] S. N. Dorogovtsev, S. N. Dorogovtsev, and J. F. Mendes, *Evolution of networks: From biological nets to the Internet and WWW*. Oxford university press, 2003.

[11] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[12] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[13] L. M. Hakik and R. El Harti, "Measuring coupling and cohesion to evaluate the quality of a remodularized software architecture result of an approach based on formal concept analysis," *International Journal of Computer Science and Network Security*, vol. 14, no. 1, pp. 11–16, 2014.

[14] W. Pan, B. Li, B. Jiang, and K. Liu, "Recode: software package refactoring via community detection in bipartite software networks," *Advances in Complex Systems*, vol. 17, no. 07n08, p. 1450006, 2014.

[15] Y. Mi, Y. Zhou, and L. Chen, "A new metric for package cohesion measurement based on complex network," in *2019 8th International Conference on Complex Networks and Their Applications*. Springer, 2019, pp. 238–249.

[16] Y. Zhou, Y. Mi, Y. Zhu, and L. Chen, "Measurement and refactoring for package structure based on complex network," *Applied Network Science*, vol. 5, no. 1, pp. 1–20, 2020.

[17] H. Abdeen, H. Sahraoui, O. Shata, N. Anquetil, and S. Ducasse, "Towards automatically improving package structure while respecting original design decisions," in *2013 20th Working Conference on Reverse Engineering*. IEEE, 2013, pp. 212–221.

[18] Amarjeet and J. K. Chhabra, "Improving package structure of object-oriented software using multi-objective optimization and weighted class connections," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 3, pp. 349–364, 2017.

[19] M. A. A. Mamun, C. Berger, and J. Hansson, "Effects of measurements on correlations of software code metrics," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2764–2818, 2019.

[20] R. Harrison, S. J. Counsell, and R. V. Nithi, "An evaluation of the mood set of object-oriented software metrics," *IEEE Transactions on Software Engineering*, vol. 24, no. 6, pp. 491–496, 1998.

[21] V. Singh and V. Bhattacherjee, "Evaluation and application of package level metrics in assessing software quality," *International Journal of Computer Applications*, vol. 58, no. 21, pp. 38–46, 2012.

[22] V. Singh and V. Bhattacherjee, "Assessing package reusability in object-oriented design," *International Journal of Software Engineering and Its Applications*, vol. 8, no. 4, pp. 75–84, 2014.

[23] C. Y. Chong and S. P. Lee, "Analyzing maintainability and reliability of object-oriented software using weighted complex network," *Journal of Systems and Software*, vol. 110, no. 1, pp. 28–53, 2015.

[24] R. C. Martin, J. Newkirk, and R. S. Koss, *Agile software development: principles, patterns, and practices*. Upper Saddle River, NJ: Prentice Hall, 2003.

[25] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

Reducing Mismatches in Syntax Coupled Hunks

Chunhua Yang[1,2], Xiufang Li[1]
[1]*School of Computer Science and Technology*
QILU University of Technology(Shandong Academy of Sciences)
[2] *Shandong Wiztek Science and Technology Co., Ltd.*
Jinan, China
jnych@126.com, lixf@qlu.edu.cn

***Abstract*—Hunks generated by textual-differencing tools are often used for understanding code changes. However, in the side-by-side view, the match between the deleted and added lines of a hunk is sometimes inconsistent with actual changes to the corresponding syntax entities. This mismatch usually occurs in syntax coupled hunks, i.e. hunks that contain changes to multiple syntax entities. It makes the hunks incomprehensible and misleading.**

A hybrid differencing algorithm is proposed to alleviate this problem. It applies tree-differencing to syntax coupled hunks to generate edits. It then maps edits back to the source code to generate adjusted hunks. Based on the current implementation, we conducted a case study on 10 open source projects. The results showed that 15% of commits contain syntax coupled hunks. And, we evaluated the results of the algorithm on 1,500 randomly drawn samples, and the correct matching rate was as high as 97%, demonstrating the effectiveness of the algorithm in reducing mismatches.

***Index Terms*—code differencing, hunks, mismatching, change understanding, software evolution**

I. Introduction

Understanding how software changes has become a regular part of modern software development. Many version management systems and IDEs provide differencing tools to present changes to the source code. The prevalent differencing tools are textual because they are efficient and not limited to programming languages. They return deltas by comparing the text values of the original and modified versions of the source code.

The common form of deltas is line-based hunks that are displayed in a unified view. For example, Fig.1 depicts two hunks returned by the well-known GnuDiff [1]. Each hunk consists of deleted lines(red), inserted lines(green), and surrounding contextual lines(white). Tools such as GitHub Diff [2], KDiff3 [3] and Mergely [4] also provide a side-by-side view, which can show the relationship between deleted lines and inserted lines more clearly. For example, Fig.2 is a split view provided by GitHub Diff to present the hunks in Fig.1.

However, the hunks generated by textual differencing tools are sometimes `syntax coupled`, that is, they contain changes to multiple syntax entities. For example, as shown in Fig.1, changes to methods *raiseTimeoutFailure* and *performOnPrimary* scattered and entangled in two hunks. This is a two-to-one method coupling, where two methods in the original version are coupled with one method in the modified version.

In syntax coupled hunks, mismatch is a common phenomenon, usually in the following two forms:

1) Mismatches between nodes in the original and modified versions of the source code. For instance, according to the hunks shown in Fig.2, the *raiseTimeoutFailure* method in the original version matches *performOnPrimary* method in the modified version. But obviously, the *performOnPrimary* method of the same name in the original and modified versions should match.
2) Mismatches between non-code lines (such as delimiters and comments), or certain elements of a signature or statement (such as arguments and annotations). For example, in Fig.2, the brace on line 543 of the original version matches the brace on line 509 of the modified version incorrectly. In fact, the two curly braces are a necessary part of the deleted If-statement and the inserted If-statement respectively, so they should not be recognized as context.

Mismatches in syntax coupled hunks hinders change understanding in the following ways:

- Mismatches between nodes can mislead users with incorrect edit operations, thereby obscuring the actual changes. For example, according to Fig.1, the edit operation between *raiseTimeoutFailure* and *performOnPrimary* is an update. But, the update should be actually between *performOnPrimary(int, ShardRouting, ClusterState)* of the original version and *performOnPrimary(int, ShardRouting)* of the modified version.
- Mismatches cause changes to the entire entity to be spread out over multiple hunks, making these hunks incomprehensible. In order to find out the actual changes, the user must go through all deleted and inserted lines.

In order to alleviate the mismatch problem in syntax coupled hunks, an algorithm is porposed in the paper. It makes the following contributions.

- It is a novel hybrid differencing algorithm for alleviating the mismatch problem. It applies tree-differencing to syntax coupled hunks to generate edit operations. It then maps edit operations back to the source code to generate adjusted hunks.

DOI reference number: 10.18293/SEKE2022-049

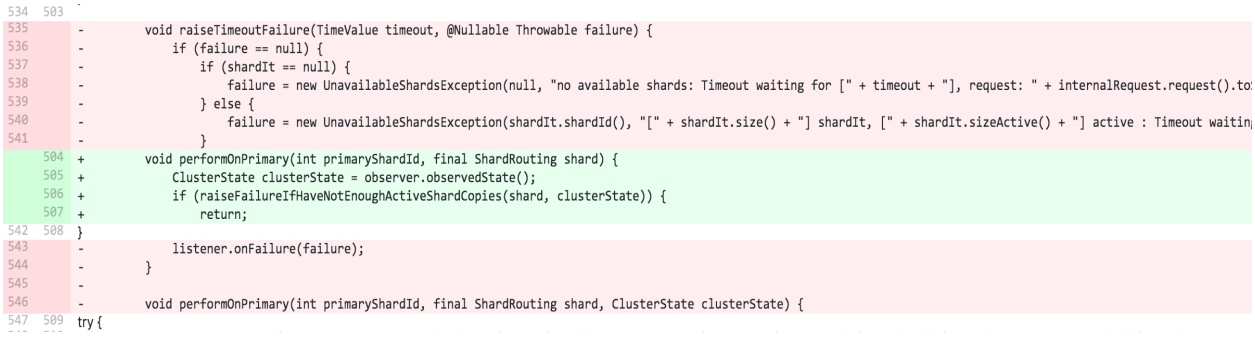


Fig. 1. An Example of Syntax Coupled Hunks.

- In addition to hunks, the algorithm outputs edit operations that facilitate analysis. For example, the hunk-level change dependency analysis becomes feasible. So far, change dependency analysis has been mainly performed at the method or line level [5] [6].

We have implemented the algorithm and conducted a case study to examine the distribution of syntax coupled hunks and evaluate the effectiveness of the algorithm.

The remainder of the paper is organized as follows. In Section 2, we present the algorithm. In Section 3, we present the implementation and case study. We review the related work in Section 4 and summarize the paper in Section 5.

II. The Algorithm

Syntactically, changes to the source code are edit operations on the nodes of the abstract syntax tree (AST). Typical edit operations include insert, delete, update, and move. If one or more hunks contain multiple edit operations on nodes on the same level of the AST, they are syntax coupled. For example, the two syntax coupled hunks in Fig.1 contain a deletion of method *raiseTimeoutFailure* and an update on method *performOnPrimary*.

We define a tuple (H, N_1, N_2) to represent syntax coupled hunks, where H is a set of hunks, N_1 and N_2 are the sets of nodes in the original and modified versions of AST whose changes occur in the hunks of H.

Inputting syntax coupled hunks, the algorithm generates the adjusted hunks through a differencing phase and a layout phase. During the differencing phase, tree-differencing is applied to the nodes in both versions to generate edit operations. Then, in the layout phase, the lines of code belonging to these edit operations are sorted, and the remaining lines are filled in the appropriate positions to produce adjusted hunks.

A. The Tree-Differencing Phase

The process is described in Algorithm 1. The algorithm inputs a hunk set H and two node sets N_1 and N_2 that belong to the original and modified versions of AST, and outputs edit operations. The main steps are as follows:

- Firstly, through the *Matching* function, the nodes in N_1 and N_2 are compared with each other to find similar nodes. (Line 1)

Algorithm 1: HASTDiff(H, N_1, N_2)

Input: a set H of hunks, two sets of nodes N_1 and N_2
Output: The set of edit operations O

1 $P \leftarrow Matching(N_1, N_2)$;
2 $Chr_1 \leftarrow \varnothing$; $Chr_2 \leftarrow \varnothing$; $O \leftarrow \varnothing$;
3 **for** *each pair* $(n_1, n_2) \in P$ **do**
4 $O \leftarrow O \cup genOp(n_1, n_2)$;
5 $C_1 \leftarrow childrenInHunks(n_1, H)$;
6 $C_2 \leftarrow childrenInHunks(n_2, H)$;
7 $H_c \leftarrow hunksCrossingNodes(C_1) \cup hunksCrossingNodes(C_2)$;
8 $U \leftarrow hunkgroupExtract(H_c, C_1, C_2)$;
9 **for** *each pair* $(H_U, N_{U1}, N_{U2}) \in U$ **do**
10 $O \leftarrow O \cup HASTDiff(H_U, N_{U1}, N_{U2})$;
11 $Chr_1 \leftarrow Chr_1 \cup unmatchedNodes(C_1, U)$;
12 $Chr_2 \leftarrow Chr_2 \cup unmatchedNodes(C_2, U)$;
13 $N_{unmatched1} \leftarrow unmatchedNodes(N_1, P)$;
14 **for** *each node* $n_1 \in N_{unmatched1}$ **do**
15 $O \leftarrow O \cup genOp(n_1, null)$;
16 $Chr_1 \leftarrow Chr_1 \cup childrenInHunks(n_1, H)$;
17 $N_{unmatched2} \leftarrow unmatchedNodes(N_2, P)$;
18 **for** *each node* $n_2 \in N_{unmatched2}$ **do**
19 $O \leftarrow O \cup genOp(null, n_2)$;
20 $Chr_2 \leftarrow Chr_2 \cup childrenInHunks(n_2, H)$;
21 $U \leftarrow hunkgroupExtract(H, Chr_1, Chr_2)$;
22 **for** *each pair* $(H_U, N_{U1}, N_{U2}) \in U$ **do**
23 $O \leftarrow O \cup HASTDiff(H_U, N_{U1}, N_{U2})$;
24 **for** *each node* $n_1 \in unmatchedNodes(Chr_1, U)$ **do**
25 $O \leftarrow O \cup genOp(n_1, null)$;
26 **for** *each node* $n_2 \in unmatchedNodes(Chr_2, U)$ **do**
27 $O \leftarrow O \cup genOp(null, n_2)$;

- Then, for each matched node pair, an *update* operation is generated. And, their children in common hunks are recursively differentiated. Their remaining children are added to the sets Chr_1 and Chr_2, respectively. (Line 3∼12)
- For each unmatched node in the set N_1 or N_2, a *delete* or

Fig. 2. The Hunks Generated by GitHub Diff for the Hunks in Fig.1.

Algorithm 2: Layout(O, H)

Input: a set O of edit operations, and a set H of hunks
Output: The set of adjusted hunks L

1 $O_{sorted} \leftarrow \varnothing$;
2 $L_1 \leftarrow sortOps(O, true)$;
3 $L_2 \leftarrow sortOps(O, false)$;
4 $insertContextRanges(L_1, H)$;
5 $insertContextRanges(L_2, H)$;
6 **while** $hasNext(L_1) \wedge hasNext(L_2)$ **do**
7 $(S_1, u_1) \leftarrow findNextUpdateOrContext(L_1)$;
8 $(S_2, u_2) \leftarrow findNextUpdateOrContext(L_2)$;
9 $O_{sorted} \leftarrow O_{sorted} \cup S_1$;
10 $O_{sorted} \leftarrow O_{sorted} \cup S_2$;
11 $addUpdateOrContext(O_{sorted}, u_1, u_2)$;
12 $addIsolatedLines(O_{sorted}, H)$;
13 $matchingIsolatedBraces(O_{sorted})$;
14 $L \leftarrow genAdjustedHunks(O_{sorted}, H)$

insert operation will be generated, and then its children will be added to Chr_1 and Chr_2 respectively. (Line 13∼20)

- A recursive differencing is applied to the children in sets Chr_1 and Chr_2. Then, a *delete* or an *insert* operation is generated for each unmatched child in Chr_1 or Chr_2, respectively. (Line 21∼27)

For syntax coupled nodes, the function *Matching* is used to determine which node in the original version is most similar to which node in the modified version. We use a strategy similar to that used in ChangeDistiller [12] to check the similarity between nodes. The function *hunkgroupExtract* is used in the algorithm to extract the syntax coupled hunk groups. And, after all edit operations are generated, we generate an *ADD* operation for consecutive *insert* operations that belong to an entire entity, and a *DEL* operation for consecutive *delete* operations that belong to an entire entity.

B. The lay out phase

In this phase, the edit operations generated in the previous phase will be laid out to generate adjusted hunks. The resulting hunks are presented in a split view.

All operations are sorted in ascending order by the starting line number of the operation. Lines of code in the original and modified versions belonging to the $update$ operation are displayed horizontally. And the update operations and context lines are set as boundaries to arrange other operations and the remaining hunk lines.

Algorithm 2 depicts the process. The main steps are as follows:

1) The operations in the set O are first sorted by the line number of the nodes in the original version(line 2) and the modified version(line 3), respectively. The sorted operations are stored in lists L_1 and L_2, respectively.
2) Then the context lines between the hunks in H are inserted into appropriate places in the lists L_1 and L_2, respectively. (Line 4∼5)
3) Next, for the lists L_1 and L_2, repeat the following steps until both lists are empty:
   a) Find the next update operation or context from each list. (Line 7∼8). The function *findNextUpdateOrContext* implements it. For a list L, the function returns the update operation or context u, and a list S of operations before u.
   b) Insert the operations of S_1 into the list O_{sorted}. Then insert the operations of S_2. (Line 9∼10).
   c) Finally, insert u_1 and u_2 into O_{sorted}, respectively. (Line 11)
4) Insert the remaining deleted or inserted lines properly into O_{sorted}. (Line 12)
   In this step, if there are isolated braces in the original and modified versions, they will be matched properly. The matching braces become the context.

5) Finally, the adjusted hunks are generated according to the lines arranged in the O_{sorted} list.

C. Illustration

Take Fig.1 as an example. Through the tree-differencing, the signature *performOnPrimary(int, ShardRouting, ClusterState)* in the original version matches the signature *performOnPrimary(int, ShardRouting)* in the modified version. And, the two children of the method *performOnPrimary* in the modified version, namely the local declaration and the If-statement, are identified as insert. Since the If-statement and its children are identified as insert operations, the changes to the If-statement are identified as an ADD.

Unmatched signature *raiseTimeoutFailure(TimeValue, Throwable)* and its children are identified as delete operations. Since its signature and its children are identified as delete operations, the method is identified as DEL.

Therefore, through the differencing phase, the following operations will be generated.

- *DEL raiseTimeoutFailure*(Line 535∼544)
- *update performOnPrimary* (Line 546 of the original version and Line 504 of the modified version)
- *ADD Local declaration* (Line 505)
- *ADD if statement* (Line 506∼508)

In the layout phase, these four operations are sorted. And the remaining isolated lines (the deleted line 545) are inserted before the update operation. The output of the algorithm is shown in Fig.3.

III. The Implementation and Case Study

We have implemented the algorithm. We use our previous work [7] to extract the syntax coupled hunk groups and use the proposed algorithm to generate the adjusted hunks in each group.

Based on the current implementation, we conducted a case study. The aim of the case study is to examine the distribution of syntax coupled hunks and evaluate the effectiveness of the algorithm.

To achieve these aims, the following research questions are to be answered:

- **RQ1:** What is the proportion of syntax coupled hunks in daily revisions?
- **RQ2:** Does the proposed algorithm generate the correctly-matched results for syntax coupled hunks?

A. The Data Set

We selected 10 open java projects from GitHub. They have different periods, stars, and scales.

The information of these projects is listed in Table I. For example, since 2010, the most popular project *elasticsearch* has 36,918 commits. Note that, we only list the number of commits containing code change hunks. Commits with only non-code changes were ignored.

TABLE I
Study Projects and Their Total Number of Revisions($Commits$) That Contain Code Change Hunks.

No.	Project	Stars	Peroid	Commits
1	activemq	1.6k	2005-12-12∼2019-11-5	5,916
2	eclipse.jdt.core	166	2001-6-5 ∼2019-11-6	15,150
3	elasticsearch	45.3k	2010-2-8 ∼2019-10-4	36,918
4	glide	27.6k	2012-12-20 ∼2019-11-6	1,625
5	guice	8.7k	2006-8-22 ∼2019-10-4	877
6	hibernate-orm	4.1k	2004-6-3 ∼2019-10-8	7,413
7	jEdit	17	2001-9-2 ∼2019-10-15	4,998
8	maven	1.9k	2003-9-1 ∼2019-11-5	5,388
9	redisson	11.1k	2013-12-22 ∼2019-11-6	2,461
10	spring-framework	33.4k	2008-7-10 ∼2019-11-5	12,930
Total				93,676

B. The Distribution of Syntax Coupled Hunks(RQ1)

For convenience, we use *hunk group*s to represent the syntax coupled hunk groups. Using the current implementation, we extracted hunk groups in each project and calculated the number of hunk groups at each granularity.

Table II lists the number of commits that contain hunk groups, the total number of hunk groups, and the number of hunk groups at each granularity for each project. We can see that 14,357 commits in the dataset contain hunk groups, which account for 15% of the total commits(as shown in Table I). In addition, the number of hunk groups with method level coupling ranks first in each project. The number of hunk groups with statement level coupling ranks second, except for Guice and Hibernate-orm. In Guice and Hibernate-orm, the class level coupling ranks second.

TABLE II
The Number of Commits With Hunk Groups Per Project, and the Number of Hunk Groups at Each Granularity. H_{class}, H_{method}, H_{field}, and H_{stmt} Represent the Number of Hunk Groups at Class, Method, Field, and Statement Granularity, Respectively.

Prj.	Commits	Number of Syntax Coupled Hunk Groups				
		Total	H_{class}	H_{mthod}	H_{field}	H_{stmt}
1	653(11%)	955	16	608	24	323
2	2,055(14%)	4,892	43	3,349	275	1,336
3	5,731(16%)	11,155	744	8,703	355	1,788
4	269(17%)	507	88	433	25	48
5	143(16%)	221	45	175	10	26
6	1,327(18%)	3,211	131	2,778	83	369
7	758(15%)	1,314	62	844	51	410
8	806(15%)	1,227	19	825	98	322
9	375(15%)	730	3	663	5	64
10	2,240(17%)	4,708	289	3,747	55	711
Total	14,357(15%)	28,920	1,440	22,125 (77%)	981	5,397 (19%)

RQ1: In summary, 15% of code change commits contain syntax coupled hunks.

C. Effectiveness of The Proposed Algorithm(RQ2)

In order to answer the second research question, based on a set of samples randomly selected from the dataset, we

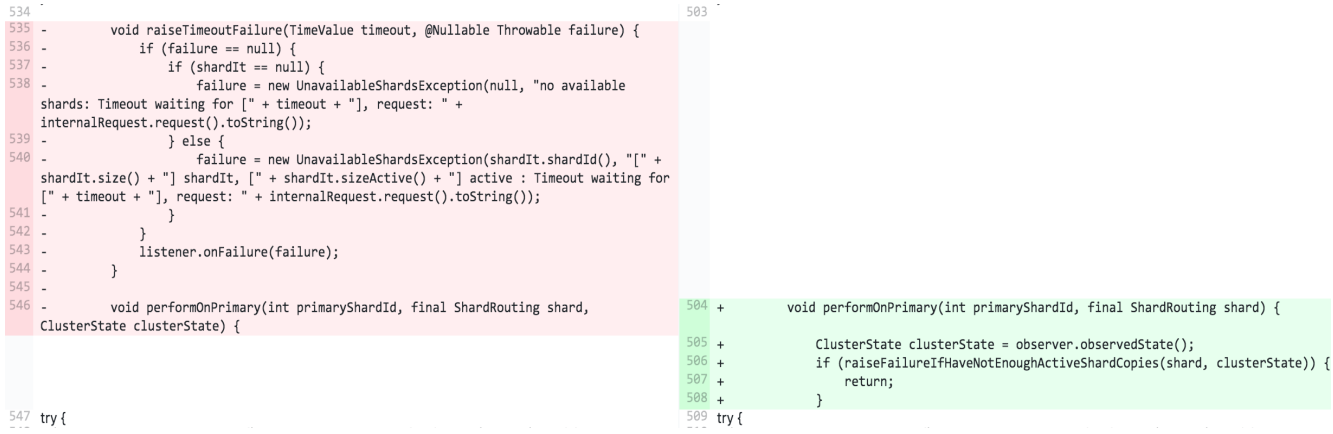


Fig. 3. The Adjusted Hunk Generated by the Proposed Algorithm for the Hunks in Fig.1.

manually evaluated the results generated by the proposed algorithm. Meanwhile, we compared the results with those generated by GitHub Diff and Mergely. We adopt these two tools due to the following reasons:

- GitHub is a platform that contains lots of open source projects. As a result, its diff is widely used for change understanding.
- Mergely provides a special side-by-side diff view style. And it provides a JS library. So, based on a hunk group set, it is easy to run Mergely on it and display the results in an html page.

For each sample hunk group, we checked the hunks generated by the algorithm, GitHub Diff and Mergely, to see if they matched correctly. We created a web page to view samples and evaluate results. The authors and three master students did the manual assessment. They have rich experience in software development. The three students first evaluated the samples of different projects independently. Then, the author reviewed their work.

We considered the coupling structure and the number of hunks contained when selecting samples. In the end, a total of 1,437 samples were selected. The left half of Table III lists the numbers of samples selected from each project. The evaluation results of the sample set are shown in the right half of table III. According to the table, the algorithm correctly matched 1,395 hunk groups, accounting for 97%. In the case of GitHub Diff and Mergely, the figures are 1,213 and 1,106, accounting for 84% and 77%, respectively.

RQ2:To sum up, in 97% of the samples, the adjusted hunks generated by the proposed algorithm are correctly matched.

The correct matching rate is higher than GitHub Diff and Mergely.

D. Threats to Validity

As the current implementation is based on Java, all selected projects are written in Java. In addition, since manual evaluation is time-consuming, the number of samples selected is not large. As a result, the validity of the case study is threatened by the programming language and sample size of the selected projects. In addition, the results of manual evaluation are influenced by inspectors. In some cases, whether one entity should be considered a match with another entity may vary from person to person.

TABLE III
THE NUMBER OF SAMPLE HUNK GROUPS SELECTED AND THE EVALUATION RESULTS. HAST REPRESENTS THE PROPOSED ALGORITHM, GH REPRESENTS GITHUBDIFF, AND MG REPRESENTS MERGELY.

Prj.	Number of Sample Hunk Groups				Correct Matching		
	H_{class}	H_{method}	H_{stmt}	**Total**	**HAST**	**GH**	**MG**
1	2	119	48	169	161	149	136
2	5	65	64	134	131	123	120
3	69	169	67	305	295	263	239
4	14	73	8	95	91	77	69
5	20	33	4	57	54	51	53
6	22	89	34	145	142	117	107
7	11	51	35	97	96	75	74
8	7	94	38	139	135	98	90
9	2	102	16	120	118	107	82
10	33	104	39	176	172	153	136
	185	899	353	1,437	1,395	1,213	1,106
					97%	84%	77%

IV. RELATED WORK

Textual Differencing. Textual-differencing tools detect text changes based on the longest common subsequence algorithm [8]. They usually generate line-based hunks(i.e., deltas). The well-known GNU diff can return added or deleted lines, which has been widely integrated into IDEs and version-control systems to calculate and present source code changes. Tools such as LDiff [9] and LHdiff [10] improve the GNU diff by detecting moved lines. To make diff easier to read, tools like GitHub Diff, Mergely and KDiff3 provide a side-by-side view and the within-line differencing to refine changes in the hunk. In [11], it is shown that Git diff with different algorithm options can give different results and revealed that the Histogram option is better for describing code changes that the default Myers option. However, for the example shown in

Fig.2, the results of the two algorithm options are the same. The mismatch problem is not improved.

Tree Differencing. Tree-differencing approaches return structural changes by comparing two ASTs representing two versions of the source code. They generate edit operations that represent changes to syntax entities.

The most famous tree-differencing algorithm is ChangeDistiller [12]. It detects changes in classes, methods, and fields.

Diff/TS [13] detected changes in various syntax granularities including classes, statements and expressions. It can detect the move actions. GumTree [14] improved ChangeDistiller by producing a shorter edit script. JSync [15] and srcDiff [16] used the longest common subsequence algorithm to compare AST nodes. MTDIFF [17] improved the move actions and generated shorter edit scripts than Gumtree, RTED, JSync, and ChangeDistiller with a higher accuracy. Higo et al. [18] considered copy-and-paste as an editing action. IJM [19] can generate more accurate move and update operations than GumTree and ChangeDistiller. CLDiff [20] aimed at generating concise linked differences. Tree-differencing approaches generate syntax edits. However, they are less efficient than textual-differencing. In [21], a hybrid method was proposed to improve GumTree matching to generate shorter edit scripts. The matching algorithm in GumTree was enhanced by using line-based textual differencing. The method is a hybrid, similar to ours. However, they focused on the optimization of GumTree, not the mismatching problem in hunks.

Tangled changes. Tao and Kim [5] proposed to partition composite code changes by grouping static related changes and methods with similar names. Herzig and Zeller [6] proposed CONFVOTERS that combine various dependencies to detect the related changes. Barnett et al. [22] proposed CLUSTERCHANGES that can relate separate regions of change by using static analysis. However, these researches focused on the tangled changes that accomplish the same task. The algorithm proposed in the paper focuses on the tangled entities in hunks.

V. Conclusions

We present an algorithm that applies tree-differencing to syntax coupled hunks to alleviate the mismatch problem. Based on current implementation, we conducted a case study to examine the distribution of syntax coupled hunks and evaluate the effectiveness of the algorithm. We found that 15% of commits contained syntax coupled hunks. And, the proposed algorithm greatly improves the mismatching in syntax coupled hunks.

Since the algorithm is based on the textual-differencing and tree-differencing is only used to compare the nodes that cross hunks, it is efficient than tree-differencing. Therefore, it can be used to extend the textual-differencing tools to reduce mismatches without worrying about efficiency. As a future work, we will provide implementations in other programming languages.

Acknowledgment

This work is supported by Shandong Provincial Natural Science Foundation (Grant No. ZR2020MF031).

References

[1] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," Communications of the ACM, 1977, vol.20, no.5, pp.350-353.
[2] "GitHub Diff," https://github.com
[3] "KDiff3," http://kdiff3.sourceforge.net/
[4] "Mergely," http://www.mergely.com/
[5] Y. Tao and S. Kim, "Partitioning composite code changes to facilitate code review," In Proceedings of the 12th Working Conference on Mining Software Repositories, 2015, pp. 180-190.
[6] K. Herzig and A. Zeller. "The impact of tangled code changes," In Proceedings of the10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, 2013, pp.121-130.
[7] C. Yang, J. Whitehead, "Pruning the AST with Hunks to Speed up Tree Differencing," In Proceedings of 26th IEEE International Conference on Software Analysis, Evolution and Reengeneering (SANER), 2019, Hangzhou, China, pp. 15-25.
[8] E. W. Myers, "AnO (ND) difference algorithm and its variations," Algorithmica, 1986, Vol.1, No.1, pp.251-266.
[9] G. Canfora, L. Cerulo and M. Di Penta, "Ldiff: An enhanced line differencing tool," In Proceedings of the 31st International Conference on Software Engineering, 2009, pp. 595-598.
[10] M. Asaduzzaman, C. K. Roy, K. A. Schneider, and M. Di Penta, " LHDiff: A language-independent hybrid approach for tracking source code lines," In 29th IEEE International Conference on Software Maintenance (ICSM 2013), 2013, pp. 230-239.
[11] Y.S. Nugroho, H. Hata, and K. Matsumoto, "How different are different diff algorithms in Git?," Empirical Software Engineering,2020, 25, pp.790-823.
[12] B. Fluri, M. Wuersch, M. PInzger, and H. Gall, "Change distilling: Tree differencing for fine-grained source code change extraction," IEEE Transactions on software engineering, 2007, vol.33, no.11, pp.725-743.
[13] M. Hashimoto and A. Mori, "Diff/TS: A Tool for Fine-Grained Structural Change Analysis," In WCRE'08: Working Conf. Reverse Eng., pages 279-288, Antwerp, Belgium, Oct. 2008.
[14] J. R. Falleri, F. Morandat, X. Blanc, M. Martinez and M. Monperrus, "Fine-grained and accurate source code differencing," In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, 2014, pp. 313-324.
[15] H. A. Nguyen, T. T. Nguyen, N. Pham, J. Al-Kofahi, and T. Nguyen, "Clone Management for Evolving Software," IEEE Trans. Softw. Eng., 38(5):1008-1026, Sep. 2012.
[16] M.J. Decker, M.L. Collard, L.G. Volkert, J.I. Maletic. "srcDiff: A syntactic differencing approach to improve the understandability of deltas," Journal of Software: Evolution and Process. 2020, 32(4).
[17] G. Dotzler and M. Philippsen, "Move-optimized source code tree differencing," In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 2016, pp. 660-671.
[18] Y. Higo, A. Ohtani, and S. Kusumoto, "Generating simpler AST edit scripts by considering copy-and-paste," In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, 2017, pp. 532-542.
[19] V. Frick, T. Grassauer, F. Beck, and M. Pinzger, "Generating Accurate and Compact Edit Scripts Using Tree Differencing," In Proceedings of the 34th IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, 23-29 Sept. 2018, Madrid, Spain, pp. 264-274.
[20] K. Huang, B. Chen, X. Peng, D. Zhou, Y. Wang, Y. Liu, and W. Zhao, "CLDIFF: Generating Concise Linked Code Differences," In Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering, Montpellier, France, 2018, pp. 679-690
[21] J. Matsumoto, Y. Higo and S. Kusumoto, "Beyond GumTree: A Hybrid Approach to Generate Edit Scripts," In Proceedings of IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), 2019, pp. 550-554.
[22] M. Barnett, C. Bird, J. Brunet, and S. K. Lahiri, "Helping developers help themselves: Automatic decomposition of code review changesets," In Proceedings of the 37th International Conference on Software Engineering, 2015, Vol. 1, pp.134-144.

Adaptive Prior-Knowledge-Assisted Function Naming Based on Multi-level Information Explorer

Lancong Liu, Shizhan Chen, Sen Chen, Guodong Fan, Zhiyong Feng, Hongyue Wu *
College of Intelligence and Computing, Tianjin University, Tianjin, China
Email: (llancong, shizhan, senchen, guodongfan, zyfeng, hongyue.wu)@tju.edu.cn

***Abstract*—Automatic function naming aims to generate a concise and meaningful name for a function, and has become a popular research area. Function naming models based on deep-learning have made significant progress in recent years. Most of the existing neural models represent a function based on the granularity of token or AST (Abstract Syntax Tree) node. However, generating function names requires more fine-grained knowledge of code, but the representation of tokens or AST nodes is not enough to capture global function semantics. In our work, we propose Apker, a novel Adaptive prior-knowledge-assisted function naming based on multi-level information explorer. The Apker includes three modules: Multi-level Information Explorer (MIE), Adaptive Prior Knowledge Adaptor (APKA) and the Generator. The MIE captures the function semantics from a local and global perspective, motivated by the understanding patterns of humans, who will first understand the meaning of each statement and then comb their logical relations to understand the whole function. The APKA uses the pre-retrieved prior knowledge to assist the model, motivated by our observation that *certain name tokens can be extracted directly from certain statements and such probability differs significantly in different types of statements*. Finally, the Generator generates function names. The experimental results demonstrate that our approach outperforms the baselines by 5.4% in Precision, 12.7% in Recall, and 7.4% in F1-score.**

***Index Terms*—function name generation, code summarization**

I. Introduction

A study has shown that developers spend more time in program comprehension than coding [13]. Developers need to provide an understandable function name because such names can help them to understand a function quickly without reading the body in detail for further information. What's more, inappropriate names may lead to software defects. For example, Abebe et al. [1] find that inconsistent identifier use contributes to the faultiness of classes. For novelties, it is difficult to generate a function name that can directly reflect their intent. Therefore, automatically generating function names becomes a critical task in reverse engineering, which can suggest appropriate names for developers.

Most existing methods use data-driven models to mine potential information from source code and then convert it into forms that can be understood easily by neural networks, e.g., some studies construct code representations from split tokens [2] and parsed AST [3, 6, 7]. Despite their effectiveness, such approaches are based on the granularity of token or AST nodes. Specifically, they input tokens or AST nodes to an encoder to learn their representations and then decode them to generate a function name. Representing source code in such way is not enough to capture global function semantics.

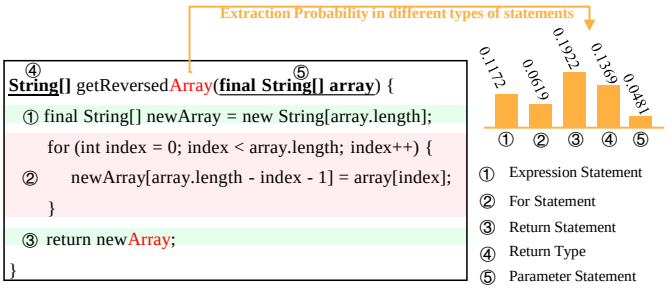


Fig. 1. Example to illustrate our motivations. We decompose the "getReversedArray" function into five statements. Each statement has its own type and we tag them respectively. The extraction probability is the prior knowledge used in the Apker.

To solve this limitation, we propose a novel approach, named Apker. There are two motivations behind the Apker. First, to mine more fine-grained knowledge of code, we follow the humans' way of understanding a function. Given a function, humans will *try to understand the meaning of each statement; then comb their logical relations to understand the whole function* [5]. Therefore, capturing the function semantics based on statement-level may be more efficient than token or AST node. Second, we observe that *certain name tokens can be extracted directly from certain statements and such probability differs significantly in different types of statements*. Such observations can be a branch to assist the model to generate name tokens correctly. Take Figure 1 as an example, there are three name tokens: "get", "reversed" and "array", among which the first two tokens don't appear in any statements, thus, they couldn't be extracted from any of them. For these tokens, the model must predict them according to its captured semantics. However, the token "array" can be extracted from the statements and it is most likely to be extracted from the "return statement" according to the prior knowledge.

To model the above working patterns, Apker introduces three modules, i.e., **M**ulti-level **I**nformation **E**xplorer (MIE), **A**daptive **P**rior **K**nowledge **A**daptor (APKA) and the Generator. The MIE captures each statement semantics from a local

DOI reference number: 10.18293/SEKE2022-085
*Corresponding Author

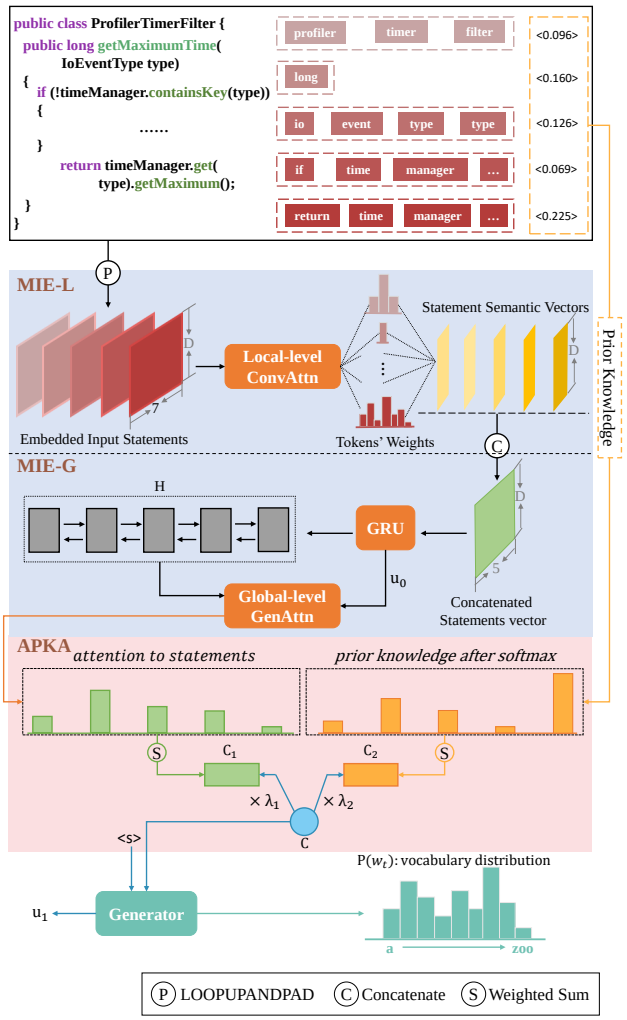


Fig. 2. Illustration of our proposed approach. This only shows the procedure of generating function names at the first time step.

perspective, based on which, it captures the whole function semantics from a global perspective. The APKA uses the prior knowledge to assist the model. Here, the prior knowledge means *The extraction probability for certain name tokens differs significantly in different types of statements*. Finally, the Generator decodes function names auto-regressively.

In summary, our main contributions are as follows:

- To explore global information by aggregating more fine-grained local information, we design the multi-level information explorer for the task of function naming.
- We fuse the prior knowledge in an adaptive way to assist the model to learn.
- The experiments and analysis on the public dataset verify the effectiveness of our approach, which is able to outperform previous state-of-the-art approaches.

II. Related Work

In this section, we introduce the related works according to the granularity of representing a function.

Early studies regard source code as plain text and employ split tokens to learn code semantics [2, 12, 14, 17, 23]. A brief history of relevant works starts with [12]. In their work, they aim to automatically generate summaries from code snippets collected from StackOverflow. They split a code snippet into tokens and apply neural machine translation networks with attention [21] to generate code summaries. Later, Allamanis et al. [2] designs a novel attention network purely based on convolutional blocks for extreme code summarization and evaluates that their network performs better than general attention.

Code is more structural than plain text. Therefore, researchers devote themselves to parsing code structure by using code analysis tools. Parsing source code into an AST is leveraged by various approaches as AST can obtain the syntax structure information [4, 5, 6, 10, 11, 15, 16, 18, 22]. At the first stage, researchers convert the ASTs into sequences before they are fed into the model. For example, deep learning similarity [22] expresses its parsed AST as a stream of the nodes by performing a pre-order visit of the sub-tree. However, converting AST to a flattened sequence destroys the original structural information. Thus, later works focus on finding a way of representing code structure directly. For example, Shido et al. [18] designs Tree-LSTM as the extension of LSTM. Tree-LSTM can learn tree structures in ASTs directly by propagating information from leaves to the root and is more effective than a sequential model used for machine translation in natural language process (NLP) when applied to source code summarization. Besides AST, other information such as data-flow and control-flow is captured [3, 7, 8].

Despite their effectiveness, all these approaches represent the source code based on the granularity of token or AST node, which is limited to capture global semantic information. What's more, they obey humans' reading habits when understanding a function.

III. The Overall Architecture of Apker

In this section, we first give a problem definition, and then introduce the three modules separately.

A. Problem Definition

The goal of this study is to create a model able to generate a function name in Java source code. As shown in Figure 2, given a function $F = \{S_0, S_1, \cdots, S_{n-1}\}$, where S_i denotes a statement, e.g., we decompose the function into five types of statements and the number behind them is the pre-counted prior knowledge, the Apker receives embedded statements as input and outputs name tokens step by step. All name tokens compose the final function name $N = \{n_0, n_1, \cdots, n_{m-1}\}$, where m is the number of tokens composing the predicted function name.

The prior knowledge of S_i is a probability and it is calculated as the number of tokens that compose the target function divided by the total number of tokens under such a statement. For example, there are 33,128,737 tokens under statements with "ClassName" type in total, among which

5,359,581 tokens exist in target function name. Therefore, the prior knowledge of statements with "ClassName" type is 0.1618. As Coganc [23] has counted the probability of 33 types of statements, we directly use it as our prior knowledge.

B. Multi-level Information Explorer (MIE)

In this paper, we introduce MIE, which includes a local-level part (MIE-L) and a global-level part (MIE-G).

MIE-L MIE-L is to capture each input statement semantics from a local perspective. As different tokens have different influences on a statement, we first use a Convolutional Attention Network (ConvAttn) [2] to weight tokens, then calculate their weighted sum and use it as the statement semantics vector. The detailed implementation is shown in Algorithm 1.

Algorithm 1 Generate a statement semantics vector

$ConvAttn(Embedded\ Statement\ E_{S_i})$
  $L_1 \leftarrow RELU(CONV1D(E_{S_i}, K_{l_1})$
  $L_2 \leftarrow CONV1D(L_1, K_{l_2})$
  $L_{feat} \leftarrow L_2/||L_2||_2$
  $L^i_{weight} \leftarrow SOFTMAX(CONV1D(L_{feat}, K))$
  $\hat{n_i} \leftarrow \sum_{j=0}^{T-1} (L^i_{weight})_j (E_{S_i})_j$
  $return\ \hat{n_i}$

Here, $L^i_{weight} = \{L^i_{weight_0}, L^i_{weight_1}, \cdots, L^i_{weight_{T-1}}\}$, where $L^i_{weight_j}$ is the jth token weight under the ith statement, and $\hat{n_i}$ is our obtained statement vector.

MIE-G MIE-G is to learn correlations between obtained statement vectors $\hat{n_i}$s and then score each statement according to its influence on the whole function semantics, which will be used for generating the final function vector. We use GRU to learn correlations in Equation (1) and employ the standard attention mechanism (GenAttn) to score statements in Equation (2).

$$\begin{aligned} CL &= Concat(\hat{n_0}, \hat{n_1} \cdots, \hat{n_{n-1}}) \\ H, u_t &= GRU(CL) \end{aligned} \tag{1}$$

where CL is the concatenated statement vectors and $H = \{h_0, h_1 \cdots, h_{n-1}\}$.

$$\alpha_{it} = \frac{exp(e(h_i, u_{t-1}))}{\sum_{i=0}^{n-1} exp(e(h_i, u_{t-1}))} \tag{2}$$

Here, $e(h_i, u_{t-1}) = a(u_{t-1}, h_i)$, a is a feedforward neural network (FNN). α_{it} denotes scores of statement S_i at time step t.

C. Adaptive Prior Knowledge Adaptor (APKA)

In fact, most of the words composing a function name can be extracted from the statements directly. Thus, when predicting a name token, if there is a thing that can assist the model to locate the statement, it will generate a correct result more quickly and correctly. This is just the functionality of the prior knowledge.

In APKA, we fuse the prior knowledge and the model in an adaptive way as shown in Equation (3). The reason we design in such way is that the model should judge if this name token can be extracted or predicted at each time step.

$$\begin{aligned} C_1 &= \sum_{i=0}^{n-1} \alpha_{it} h_i \\ P &= SOFTMAX(p_0, p_1, \cdots, p_{n-1}) \\ C_2 &= \sum_{i=0}^{n-1} P h_i \\ \lambda_1, \lambda_2 &= \sigma(u_t W_u + C_1 W_{C_1} + C_2 W_{C_2}) \\ C &= \lambda_1 C_1 + \lambda_2 C_2 \end{aligned} \tag{3}$$

Here, C_1 and C_2 are context vectors generated by statement attentions and the prior knowledge respectively. p_i is the prior knowledge of S_i. W_u, W_{C_1} and W_{C_2} are learnable parameters. The computed λ_1, $\lambda_2 \in [0, 1]$ weight the expected importance of C_1 and C_2 respectively and their values can show the adaptability.

D. Generator

Generator aims to decode the final function vector C into an actual function name. For each step t, we predict the tth word by generating the vocabulary distribution. Generator takes the current input word $x_t (x_0 =< s >)$, last hidden state u_{t-1} and C as input. We describe it with Algorithm 2.

Algorithm 2 Generate vocabulary distribution of each step t

$Generator(x_t, u_{t-1}, C)$
  $e_{x_t} \leftarrow Embed(x_t)$
  $input \leftarrow Concat(e_{x_t}, C)$
  $output, s_t \leftarrow GRU(input, u_{t-1})$
  $output \leftarrow Linear(Concat(output, C))$
  $P_{w_t} \leftarrow SOFTMAX(output)$
  $return\ P_{w_t}, u_t$

During training, the overall loss for the whole sequence is calculated as the average loss at each time step shown in Equation (4), which is the negative log likelihood of the word w_t for that step:

$$loss = \frac{1}{T} \sum_{t=0}^{T} (-logP(w_t)) \tag{4}$$

IV. Experiments

In this section, we firstly describe used dataset as well as some widely-used metrics and experimental settings in detail. Then we present the evaluation and analysis of the proposed approach.

A. Datasets, Metrics and Settings

Dataset We use the Java dataset collected by [2], which is obtained from 11 open-source Java projects on GitHub. This dataset has been split into training/testing/validation by projects. Particularly, we remove the functions whose length of their names is less than 2 or longer than 6 because

TABLE I
COMPARISONS OF OUR APPROACH AND THE EXISTING METHODS

Approaches	Metrics			Used Code Form					
	Precison↑	Recall↑	F1↑	tokens	AST	DE-AST	CFG	AST path	stmts
Cognac	0.671	0.597	0.632	√					
ConvNet	0.459	0.394	0.406	√					
TBCNN	0.409	0.318	0.355		√				
TreeCaps	0.526	0.414	0.468		√				
GGNN	0.403	0.353	0.369			√			
GREAT	0.473	0.400	0.436	√		√			
Sequence GINN	0.648	0.562	0.602				√		
Code2vec	0.234	0.220	0.214					√	
Code2seq	0.504	0.354	0.426	√				√	
Apker	**0.701**	**0.673**	**0.679**	√					√

TABLE II
EFFECTIVENESS OF MIE.

Attention type	ConvAttn	GenAttn	APKA	Metrics		
				Precision	**Recall**	**F1**
Basic				0.469	0.411	0.431
+ConvAttn	√			0.609	0.556	0.573
+GenAttn		√		0.553	0.504	0.520
MIE	√	√		**0.666**	**0.628**	**0.638**

such functions are not practical. Our dataset contains 163,168 functions finally.

Metrics We measure prediction performance using Precision, Recall, and F1 over the sub-words in generated names, following the metrics used by [5, 6, 7].

Settings To train our model, we optimize the objective using stochastic gradient descent with RMSProp [20] and Nesterov momentum [9]. We use dropout [19] on all parameters and gradient clipping. Each of the parameters in the model is initialized with normal random noise around zero, except for W_u, W_{C_1}, and W_{C_2} in APKA are initialized with kaiming normal. For ConvAttn, the best values for k_1, k_2, w_1, w_2, and w_3 are 8, 8, 24, and 29. The embed and hidden dimensions are set to 100. The dropout rate is set to 25%. The batch size is set to 32.

B. Comparison with State-of-the-art Methods

We compare our approach with a wide range of state-of-the-art function naming models, i.e., Cognac [23], ConvNet [2], TBCNN [16], TreeCaps [6], GGNN [3], GREAT [8], Sequence GINN [24], Code2Vec [5] and Code2Seq [4]. As shown in Table I, our Apker outperforms state-of-the-art approaches across all metrics. Compared to these best scores of different metrics, our method has a Precision improvement of 5.4%, a Recall improvement of 12.7% and an F1-score improvement of 7.4%. The improved performance demonstrates the validity of our approach.

C. Effect of MIE

ConvAttn and GenAttn are the two key components to implement the MIE module. Therefore, to evaluate the effectiveness of MIE, we make an ablation study.

Here, we construct four baseline networks. Note that we all remove the APKA module in these four networks. The first one (denoted as "Basic") is to remove the two attention mechanisms. The second one ("+ConvAttn") employs the original MIE-L to weight token attentions for each statement, while the third one ("+GenAttn") uses the same MIE-L as the first one but adds GenAttn after getting final statement vectors. The last baseline network ("MIE") is our approach without APKA.

Table II shows the effectiveness of MIE. From this table, we can know that both "+ConvAttn" and "+GenAttn" outperform "Basic" on all metrics and "+ConvAttn" has a significant improvement compared to "+GenAttn". It indicates that the generated statement vector considering different weights of tokens reflects the statement semantics better. Moreover, MIE outperforms "+ConvAttn" for all different evaluation metrics. These results show that our proposed MIE makes significant contributions to accurate function name generation.

TABLE III
EFFECTIVENESS OF APKA.

Measure	MIE	+SPKA	+APKA
Precision↑	0.666	0.682	**0.701**
Recall↑	0.628	0.659	**0.673**
F1-score↑	0.638	0.663	**0.679**

D. Effect of APKA

In order to evaluate the performance of APKA, we implement a static prior knowledge adaptor named SPKA to make comparisons in Equation (5). The difference between SPKA and APKA lies in the way to fuse the prior knowledge.

$$\begin{aligned} \alpha_{it} &= \alpha_{it} + P \\ C &= \sum_{i=0}^{n-1} \alpha_{it} h_i \end{aligned} \tag{5}$$

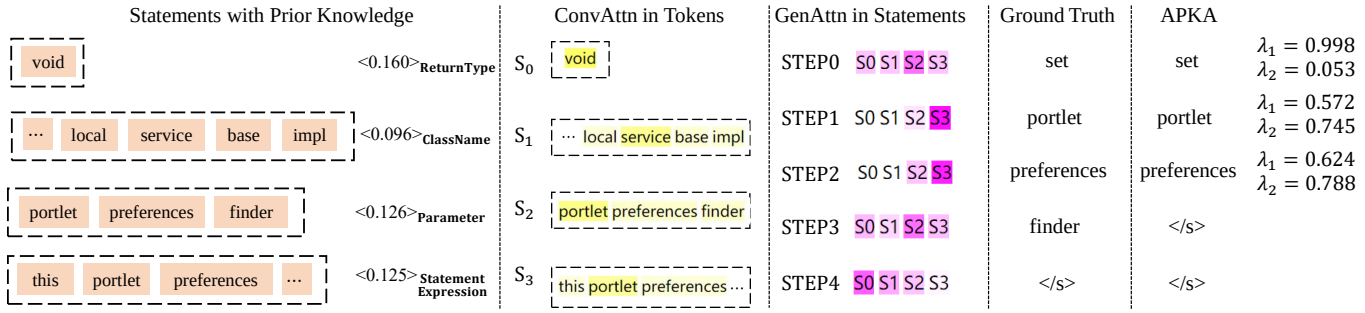


Fig. 3. We give the visualization of our approach. In the *Statements with Prior knowledge* step, each dashed box refers to a statement and it has the prior knowledge behind. S_i refers to the ith statement. Tokens highlighted in yellow and statements highlighted in purple represent their attention scores. All colors follow a principle: the darker a color is, the more important an item is. Each pair of λ_1 and λ_2 denotes the weights of the model and the prior knowledge at each time step respectively.

As shown in Table III, it is clear that our APKA successfully boosts the performance, verifying its effectiveness. Adding the prior knowledge to the statement attentions directly means the totally same extraction probability for all time steps, ignoring the dynamic probability to extract or to predict at each time step.

E. Visual Analysis

In Figure 3, we give an illustrative example to better understand our approach. As we can see, in Apker, the ConvAttn first learns statement semantics by scoring their inner tokens, e.g., the first statement S_0 only has a token "void", thus ConvAttn scores it high; S_1 attends more to the token "service" and the last two statements both attend more to the token "portlet" than others. Then the GenAttn scores these statements according to their influence on the whole function semantics.

Now, look at how APKA takes effect. As the name token "set" doesn't appear in any statement, the APKA is expected to weigh more in the model rather than the prior knowledge. As we can see, the λ_1 and λ_2 are 0.998 and 0.053 respectively, satisfying our expectations. But When generating "portlet" and "preferences", the APKA should attend to the prior knowledge because the two tokens can be extracted from the last two statements directly. Their λ_1s and λ_2s are also corresponding to our expectations. Though the last name token "finder" is not predicted rightly, the presented result can still verify the capability of Apker to generate function names.

V. Discussion

In this section, we discuss the strength and some threats of Apker.

A. Strength of Apker

We have evaluated three main advantages of Apker that may explain its effectiveness in function naming: (a) *A more comprehensive representation of source code.* The Apker decomposes a function into several types of statements. The experiments also validate the capability of statement-level approach than token or AST nodes. (b) *A multi-level attention mechanism.* The Apker uses a multi-level attention mechanism, among which, local-level attention infers the contribution of each token to the statement and global-level infers the contribution of each statement to the whole function.

B. Threats to Validity and Limitations

Our proposed Apker may suffer from two threats. One threat is on the prior knowledge. The obtained statement prior knowledge is highly dependent on the empirical study. The correctness of the prior knowledge has an influence on the predicted results.

Another threat lies in the extensibility of Apker. Our model needs to identify different types of statements based on a static analysis tool. There are many tools to analyze Java source code but few in other programming languages. Therefore, it may be difficult to extend our approach to other languages. Besides, we also consider the class name of the target function and other statements in caller/callee functions. However, they can only be parsed from a whole program. Therefore, it is challenging to extend Apker to some datasets collected from other channels where only a small code snippet can be extracted.

VI. Conclusion

In this paper, we proposed a novel prior-knowledge-guided neural network for the task of function naming. It includes a multi-level information explorer to capture local information with learning to aggregate them to global information and an adaptive prior knowledge adaptor. Two attention mechanisms (ConvAttn and GenAttn) were used so that different contributions of tokens to a statement and statements to a function can be inferred. In particular, we fuse prior knowledge in Apker to assist the model. We demonstrate the superior performance of the proposed framework. For future work, we plan to look into more accurate and valuable prior knowledge. Furthermore, we will conduct comprehensive experiments on other programming languages such as Python and C.

VII. Acknowledgement

This work is supported by the National Natural Science Key Foundation of China grant No.61832014 and No.62032016, National Natural Science Foundation of China grant No.

62102281, and the Natural Science Foundation of Tianjin City grant No.19JCQNJC00200.

REFERENCES

[1] Surafel Lemma Abebe, Venera Arnaoudova, Paolo Tonella, Giuliano Antoniol, and Yann-Gael Gueheneuc. Can lexicon bad smells improve fault prediction? In *2012 19th Working Conference on Reverse Engineering*, pages 235–244. IEEE, 2012.

[2] Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. In *International conference on machine learning*, pages 2091–2100. PMLR, 2016.

[3] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*, 2017.

[4] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. *arXiv preprint arXiv:1808.01400*, 2018.

[5] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[6] Nghi DQ Bui, Yijun Yu, and Lingxiao Jiang. Treecaps: Tree-based capsule networks for source code processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 30–38, 2021.

[7] Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. Structured neural summarization. *arXiv preprint arXiv:1811.01824*, 2018.

[8] Vincent J Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, and David Bieber. Global relational models of source code. In *International conference on learning representations*, 2019.

[9] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. Neural networks for machine learning. *Coursera, video lectures*, 264(1):2146–2153, 2012.

[10] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE, 2018.

[11] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation with hybrid lexical and syntactical information. *Empirical Software Engineering*, 25(3):2179–2217, 2020.

[12] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, 2016.

[13] Andrew J Ko, Brad A Myers, Michael J Coblenz, and Htet Htet Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on software engineering*, 32(12):971–987, 2006.

[14] Yi Li, Shaohua Wang, and Tien N Nguyen. A context-based automated approach for method name consistency checking and suggestion. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 574–586. IEEE, 2021.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[16] Lili Mou, Ge Li, Zhi Jin, Lu Zhang, and Tao Wang. Tbcnn: A tree-based convolutional neural network for programming language processing. *arXiv preprint arXiv:1409.5718*, 2014.

[17] Son Nguyen, Hung Phan, Trinh Le, and Tien N Nguyen. Suggesting natural method names to check name consistencies. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 1372–1384, 2020.

[18] Yusuke Shido, Yasuaki Kobayashi, Akihiro Yamamoto, Atsushi Miyamoto, and Tadayuki Matsumura. Automatic source code summarization with extended tree-lstm. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[20] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[22] Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. Deep learning similarities from different representations of source code. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 542–553. IEEE, 2018.

[23] Shangwen Wang, Ming Wen, Bo Lin, and Xiaoguang Mao. Lightweight global and local contexts guided method name recommendation with prior knowledge. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 741–753, 2021.

[24] Yu Wang, Ke Wang, Fengjuan Gao, and Linzhang Wang. Learning semantic program embeddings with graph interval neural network. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–27, 2020.

Towards Accurate Knowledge Transfer between Transformer-based Models for Code Summarization

Chaochen Shi [1], Yong Xiang [2], Jiangshan Yu [3], and Longxiang Gao [4]
[1, 2] School of Information Technology, Deakin University, Australia
[3] Faculty of Information Technology, Monash University, Australia
[4] Qilu University of Technology (Shandong Academy of Sciences), China
Email: {shicha, yong.xiang}@deakin.edu.au, j.yu.research@gmail.com, longx.gao@gmail.com

***Abstract*—Automatic code summarization generates high-level natural language descriptions of code snippets, which can benefit software maintenance and code comprehension. Recently, Transformer-based models achieved state-of-the-art performance on code summarization tasks. However, there are data gaps in neural model training for some programming languages. To fill this gap, we propose a novel transfer learning approach to accurately transfer knowledge between Transformer-based models. We train a discriminator to identify which heads of the multi-head attention module should be transferred. On this basis, we define a transfer strategy of parameter matrices. We evaluated the proposed transfer learning approach on four state-of-the-art Transformer-based code summarization models. Experimental results show that models with transferred knowledge outperform original models up to 10.70% in BLEU, 5.36% in ROUGE-L, and 4.34% in METEOR.**

Keywords-Transfer Learning; Code Summarization

I. INTRODUCTION

Automatic code summarization is a seq2seq (sequence to sequence) task of automatically generating natural sentences to describe a code snippet. Due to the general lack of high-quality source code comments in software development, automatic code summarization tools are of great significance in helping developers understand code and improve development efficiency. In recent years, due to the outstanding performance of Transformer [7] on seq2seq tasks, many studies on automatic code summarization leverage Transformer-based architectures [1, 2, 5, 8]. These state-of-the-art studies have proven that Transformer architecture is highly effective in capturing dependencies between code tokens. However, such neural code summarization models require a large number of <*code, comment*> pairs as ground truth data for training. For mainstream programming languages such as Java and Python, there are already large public datasets such as SIT [3] and CodeSearchNet [4] with millions of records. However, there is a lack of available large corpus for programming languages with smaller communities, such as Solidity that specializes in smart contract development on blockchain platforms. It leads to the poor performance of directly training neural models on such programming languages.

A potential solution to the gaps mentioned above is to transfer specific knowledge from well-trained code summarization models to less-trained models through transfer learning. The idea comes from the intuition that shared features (types, syntax, object-oriented characteristics, etc.) exist in different programming languages. As for Transformer, each parallel attention layer (head) pays attention to an individual subspace, which means that some heads may focus on shared features among different programming languages. This paper proposes a novel transfer learning approach to accurately transfer the Transformer's multi-head attention between programming languages. We train a discriminator *D* to filter heads that focus on similar features in transform-based models of different programming languages. Then we transfer the corresponding model parameters from the source domain to the target domain according to the similarity weights of heads. Next, we train the model of the target domain to fit its dataset. Our contributions are as follows:

- We propose the transfer learning approach dedicated to conducting accurate knowledge transfer between Transformer-based code summarization models, which can improve the performance of models in the target domain and speed up the training process.
- Taking Python and Solidity as examples, we compare the performance with and without the proposed transfer learning approach on four state-of-the-art Transformer-based models. The experimental results demonstrate the maximum improvement reaches 10.70%, 5.36%, and 4.34% in BLEU, ROUGE-L, and METEOR, respectively.

II. BACKGROUND AND RELATED WORK

Attention mechanism has become an integral part of sequence modeling, allowing the model to select the more critical information to the current task. The self-attention mechanism is a variant of the attention mechanism, which reduces the dependence on external information and better captures the internal correlations of data or features. The attention is calculated as equation 1.

$$Attn(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Where *Q, K, V* are three tensors of the input, d_k is the dimension of *Q* and *K*. In self-attention mechanism, *Q*=*K*=*V*.

DOI reference number: 10.18293/SEKE2022-111

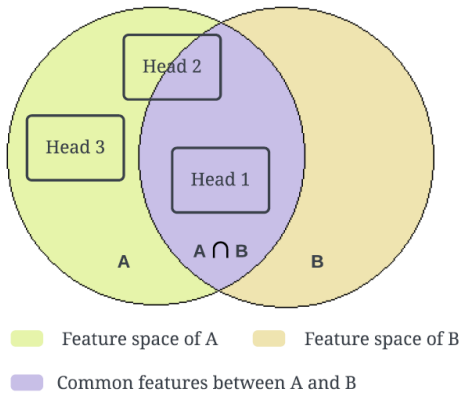


Figure 1. The feature space covered by different heads. Apparently, head 1 covers more shared features between A and B than head 2, 3.

Transformer is an encoder-decoder model relying on attention mechanism as the main component, which enables parallel computing and improves the feature extraction ability. As the state-of-the-art solutions to code summarization tasks, Transformer-based models have been widely studied. Ahmad [1] used Transformers to extract textual and structural features from code token sequence and AST (Abstract Syntax Tree), respectively. Clement [2] proposed Pymt5, a Python method text-to-text transfer Transformer, which is trained to translate between all pairs of Python method feature combinations. Liu [5] proposed a joint summary generation model based on improving Transformer, adding pointer mechanism and consistency loss function to keep the original meaning in generated sentences as much as possible. Wang [8] designed a structure encoding algorithm to represent hierarchical code structures. They combined it with BERT (a Transformer-based pre-trained model) to better extract code structural features. These mentioned works show Transformer-based models outperform existing RNN- and LSTM-based models by a large margin, playing the leading role in code summarization tasks recently.

III. Our Approach

The knowledge mentioned in this paper is the ability of the model to extract features from the input. Supposing there are two programming languages *A* and *B*, where A has a well-trained transformer-based code summarization model M_A while *B* has a structurally identical but undertrained model M_B. Our target is accurately transferring knowledge from M_A to M_B to facilitate the training process of M_B. Our research questions include:

- RQ1: How to identify the knowledge (heads) we need to transfer?
- RQ2: How to transfer the selected knowledge between Transformer-based models?

We address RQ1 and RQ2 in section III-A and III-B, respectively.

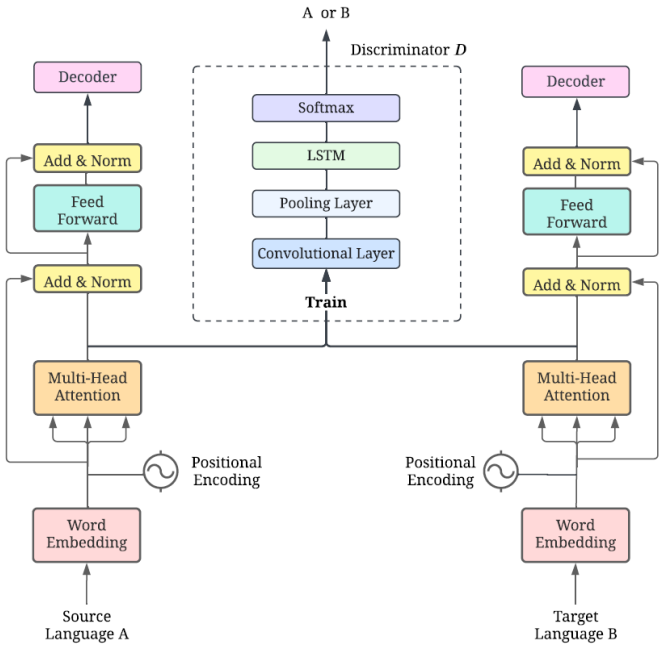


Figure 2. The structure and training process of the discriminator D. The left and right half are of the same Transformer M_A trained by source language A. D receives data from the multi-head attention module and tries to identify whether inputs are from language *A* or *B*.

A. Transferable Knowledge Identification

To address RQ1, we need a strategy to identify transferable knowledge which fits both source domain and target domain. For Transformer-based architectures with *h* heads, the model learns vector representations $z_1, z_2, \dots, z_h$ from each head. Then the multi-head attention representation Z is calculated as equation 2.

$$MultiAttnZ = Concat(z_1, z_2, \dots, z_h)W^O \quad (2)$$

where

$$z_i = Attn\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (3)$$

Where W_i^Q, W_i^K, W_i^V and W^O are parameter matrices of linear layers as Fig. 3 shows. The multi-head attention module of Transformer allows the model to jointly attend to information from *h* different representation subspaces at different positions [7]. As Fig.1 shows, some heads may focus on more shared features between *A* and *B* than others, which means such heads can contribute more transferable knowledge. We call them transferable heads.

Supposing we already have a well-trained Transformer-based model M_A for *A*. We train a discriminator *D* to identify transferable heads in M_A. As Fig. 2 shows, *D* can be regarded as a binary classification model which receives a code representation *Z* of M_A 's multi-head attention module and tries to identify whether *Z* is from *A* or *B*. Since the code token sequence is generally long, we use a Convolutional layer and a Pooling layer (average pooling) to reduce the dimensionality of the input. Then the input goes through an LSTM layer and a Softmax layer, outputs the probability *P* of being classified into

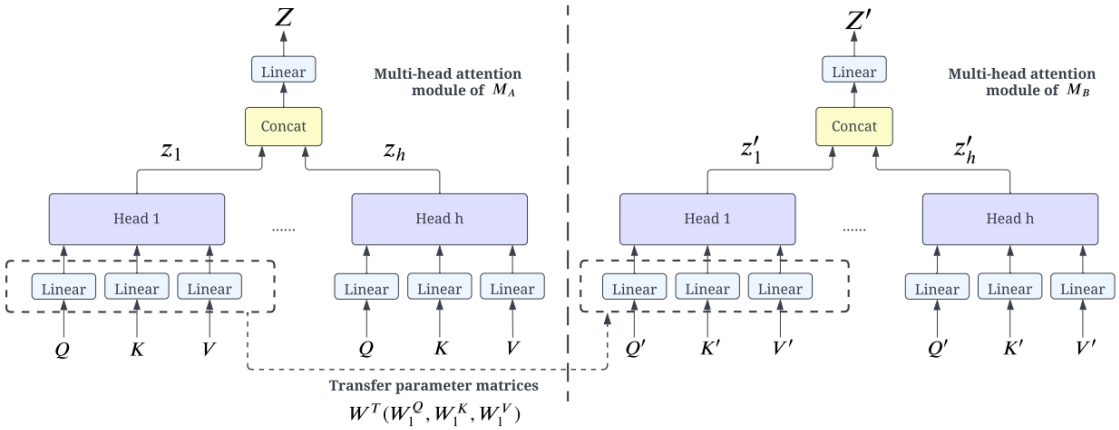


Figure 3. The knowledge transfer process. Taking head 1 as the transferable head, its parameter matrices of linear layers are transferred from M_A to M_B.

the target class. The classification result is based on *P* with a threshold of 0.5. The training target is to minimize the loss as equation 4.

$$Loss = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})) \quad (4)$$

Where $\hat{y}$ is the probability that the model predicts the sample as a positive example; y is the label of the sample, which takes a value of 1 if the sample is positive, and 0 otherwise.

The trained *D* identifies which programming language *Z* is from. The rationale is that *D* captures the non-shared features between *A* and *B*. Since the output of multi-head attention comes from the weighted concatenate of each head, these non-shared features also come from the subspaces of heads. Non-transferable heads contribute more to non-shared features, helping *D* distinguish *A* from *B*; Transferable heads are the opposite. We use the F1 score to represent the predictive performance of *D* since it considers both precision and recall.

$$F1 = \frac{2Percision \cdot Recall}{Percision + Recall} \quad (5)$$

To evaluate the contribution of each head to *D*, we replace the concatenate of *h* different heads as *h* repeated heads in the multi-head attention module. Then the *Z* becomes Z_i for the *i*-th head as equation 6.

$$Z_i = Concat\left(\overbrace{z_i, z_i, \dots, z_i}^{h}\right) W^O \quad (6)$$

B. Transfer Process

In this way, the feature space of Z_i is totally contributed by the *i*-th head. Since *Z* has the same size as *Z*, it can be directly predicted by *D*. The strategy of selecting transferable heads is based on the F1 score of each head. Head with a lower F1 score means its output Z_i is more difficult to be predicted by *D*, i.e., this head more focuses on shared features. In this paper, we regard the F1 score of the multi-head module as a threshold: For heads with a lower F1 score than the threshold, we identify them as transferable heads.

The knowledge transfer process shows as Fig. 3. According to equation 1 and 3, the output z_i of the *i*-th head is calculated by linear transformations of input *Q*, *K*, *V*. Thus, the knowledge transfer is to transfer the parameter matrices W_i^Q, W_i^K, W_i^V of linear layers. Supposing there is an undertrained multi-head attention module of *B*, we transfer the linear layer of the *i*-th transferable head from *A* to *B*. Then the output z'_i of the transferred head is

$$z'_i = Attn\left(QW_i^Q, KW_i^K, VW_i^V\right)t_i \quad (7)$$

Where t_i is the transfer weight normalized from the F1 score of the *i*-th head.

After M_B receives *n* transferable heads from M_A, M_B is able to capture some common features between *A* and *B* as well. However, there are domain-specific features of *B* that need to be captured by M_B to fit the code summarization task in its domain. Thus, we freeze the transferred heads and train other components (W_i^O, feed-forward layers, and decoder) of M_B on its dataset. Supposing there are *n* transferred heads, the remaining *h* - *n* heads are trained from scratch to capture domain-specific features of *B*.

IV. Experiments and Discussion

A. Dataset and Baselines

We take Python as the source domain and Solidity as the target domain in our experiments. These two are both object-oriented programming languages, while Solidity is used in the blockchain area. Their shared features (types, object-oriented characteristics, etc.) and non-shared domain-specific features are significant, which is ideal for transfer learning.

We randomly selected 450K Python <*code*, *comment*> pairs from the widely used corpus CodeSearchNet [4], and 45K Solidity pairs from the corpus used in [6] to build our dataset. We split original code texts by a set of symbols, i.e., { . ,"~' : * () ! - (space)}. Each token written in camel-case or snake-case shall be segmented into the original word. For example, the token **helloWorld** or **hello_world** shall be segmented into two separate words: **hello** and **world**. On this basis, we build our vocabularies for sentence generation. We use four state-of-the-art Transformer-based code summarization model [1, 2, 5, 8] introduced in section II as baselines.

TABLE I. PERFORMANCE COMPARISON (IN PERCENTAGE) OF BASELINES WITH AND WITHOUT TRANSFER LEARNING.

Language	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Rouge-L	METEOR
Python	[1]	43.31	38.54	35.25	31.02	39.26	17.17
	[2]	40.17	35.42	31.09	26.34	33.67	14.18
	[5]	41.05	36.22	33.02	28.39	34.15	15.38
	[8]	39.39	35.75	34.87	27.69	38.66	16.54
Solidity	[1]	27.31	23.13	19.65	14.50	21.09	9.96
	[1]+TL	**33.50**	**30.37**	**28.26**	**25.20**	**26.45**	**13.19**
	[2]	24.49	21.72	18.37	12.66	18.63	7.09
	[2]+TL	**30.56**	**26.21**	**23.33**	**20.01**	**21.45**	**11.43**
	[5]	24.99	22.37	19.45	13.78	18.56	8.34
	[5]+TL	**31.87**	**28.08**	**23.76**	**18.44**	**21.96**	**10.85**
	[8]	22.18	17.46	14.12	10.49	17.30	9.24
	[8]+TL	**31.57**	**26.35**	**23.88**	**19.29**	**20.96**	**11.02**

a. Python is the source domain, and Solidity is the target domain. TL is the abbreviation of the proposed transfer learning approach.

B. Experimental Settings and Metrics

The word embedding size and multi-head attention size are both set to be 512. The number of heads *h* is set to be 16. The mini-batch size is set to be 64 with a learning rate of 0.001 for both Transformer-based models and the discriminator. We use 10-fold cross-validation for experiments and run 10 epochs in the training processes. All the experiments in this paper are implemented with Python 3.7 and run on Google Colab with an NVIDIA Tesla P100 GPU.

We evaluate the performance of code summarization task based on three widely used metrics, BLEU, ROUGE-L, and METEOR. These three objective metrics are close to human evaluation criteria. BLEU is obtained by calculating the n-gram matches between the candidate and reference sentences. We use BLEU 1-4 as our metrics as in [8]. ROUGE-L is a metric that matches the longest common sequence between two sentences and returns the recall rate. METEOR combines both uni-gram matching precision and recall rate using harmonic mean.

C. Experimental Results

Table I compares the effectiveness of proposed transfer learning approach between four baselines on our dataset. The transfer learning mechanism brings improvement of BLEU 1-4 ranged from 4.31% (BLEU-3 of [5]) to 10.70% (BLEU-4 of [1]); For ROUGE-L, the range of improvement is from 2.82% ([2]) to 5.36% ([1]); For METEOR, the range of improvement is from 1.78% ([8]) to 4.34% ([2]). Overall, the performance of the four baselines in the target domain has been significantly improved after leveraging transfer learning.

D. Threats to Validity

Similarity between programming languages. Transfer learning requires a high similarity between the data features of the source and target domains. Both Python and Solidity used in our experiments are object-oriented languages, so it is adequate to use transfer learning. It is not applicable to transfer knowledge between programming languages with significant differences in structure and syntax, such as an object-oriented language and an assembly language.

Number of transferable heads. Here is a trade-off: Fewer transferable heads mean less knowledge would be transferred; More transferable heads would reduce the model's ability to learn domain-specific features because there would be fewer trainable heads. For models with different numbers of heads, the ideal threshold of identifying transferable heads may vary according to experimental results.

V. CONCLUSION

We propose a transfer learning approach to accurately transfer knowledge between Transformer-based models for code summarization tasks. We train a discriminator to identify transferable heads that focus more on common features between source and target domains. Our approach only transfers knowledge in similar feature spaces between domains, which is more adaptive than simply copying and freezing neural layers. We conducted experiments on a dataset built from a large public corpus, proving the effectiveness of our approach.

REFERENCES

[1] Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. A transformer-based approach for source code summarization. arXiv preprint arXiv:2005.00653, 2020.

[2] Colin B Clement, Dawn Drain, Jonathan Timcheck, Alexey Svyatkovskiy, and Neel Sundaresan. Pymt5: multi-mode translation of natural language and python code with transformers. arXiv preprint arXiv:2010.03150, 2020.

[3] Wu Hongqiu, Zhao Hai, and Zhang Min. Code summarization with structure-induced transformer. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL), 2021.

[4] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. arXiv preprint arXiv:1909.09436, 2019.

[5] Xin Liu and Liutong Xu. A combined model for extractive and abstractive summarization based on transformer model. In SEKE, pages 396–399, 2020.

[6] Chaochen Shi, Y ong Xiang, Jiangshan Y u, and Longxiang Gao. Semantic code search for smart contracts. arXiv preprint arXiv:2111.14139, 2021.

[7] Ashish V aswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[8] Ruyun Wang, Hanwen Zhang, Guoliang Lu, Lei Lyu, and Chen Lyu. Fret: Functional reinforced transformer with bert for code summarization. IEEE Access, 8:135591–135604, 2020.

Towards Lightweight Detection of Design Patterns in Source Code

Jeffy Jahfar Poozhithara Hazeline U. Asuncion Brent Lagesse
University of Washington Bothell, WA, USA
E-mail: jeffyj@uw.edu, hazeline@uw.edu, lagesse@uw.edu

Abstract

Identifying which design patterns exist in source code helps maintenance engineers better understand source code and determine if new requirements can be satisfied. Automated techniques for finding design patterns generally require much time to label training datasets or to specify rules/queries for each pattern, and is difficult to extend support to secure design patterns (SDPs) and combination patterns. To address these challenges, we introduce PatternScout, a technique for automatic generation of SPARQL queries from UML Class diagrams and Sequence diagrams. These queries are used to detect patterns in the source code. Our results indicate that PatternScout can detect object-oriented design patterns (OODP) with accuracy that is comparable or better than existing techniques. It can also generate queries for SDPs that can be represented as UML Class diagrams.

1 Introduction

Since design patterns assist with satisfying security requirements, it is important for maintenance engineers to determine which patterns already exist in the code, including SDPs. Finding design patterns can be time-consuming, due to manual work required to reverse engineer the code [1]. Automated techniques also have challenges. Mining techniques involve the time-consuming task of manual labeling the training dataset and manual checking results due to false positives [2]. On the other hand, detection using rules [3] and queries [4] also involve time-consuming specification of design patterns and suffer from false negative results, as they generally lack the flexibility to handle variations. This challenge is especially pronounced in SDPs, which have a higher level of variability than OODPs [5].

Meanwhile, Semantic Web technologies provide a means of encapsulating rich information, using a graph known as Resource Description Framework (RDF). When source code is represented as an RDF graph, we can see design- and code-level concepts (e.g., class relationships, class properties), which are not captured in other graph representations of source code (e.g., abstract syntax trees). There are several query technologies to extract information from RDF; among these SPARQL is the most popularly used [6]. Running queries on an RDF yields highly accurate results, as these only retrieve results with matching graph pattern. However, as we mention above, this technique also suffers from false negatives, as the results obtained are very specific. Furthermore, using Semantic Web query languages, such as SPARQL, is difficult because the user needs to learn not only the query language syntax, but also the vocabulary and relationships in the data [6]. This is why researchers developed automated techniques for generating SPARQL queries [7]. However, none of these techniques cater to software.

To address these challenges, we developed PatternScout which takes advantage of Semantic Web technologies while overcoming its difficulties. First, the difficulty of using SPARQL queries is handled by automatically generating queries from UML Class and Sequence diagrams. This approach works as repositories of common design patterns exist [8, 9] and they already include UML Class diagrams in their descriptions. This also applies to SDPs as many of them also include Class diagrams [10]. Second, we overcome the limitation of low recall by creating a catalogue of known design pattern variations [11] and checking for these variations in the source code.

Our main contribution is a novel technique to generate SPARQL queries that can correctly identify design patterns. Compared to other methods, our approach is more lightweight because it does not involve any manual training and does not require manually defining rules or queries. In addition to the 23 GoF patterns supported by state-of-the-art design pattern detection techniques, PatternScout can generate SPARQL queries for secure design patterns, object-oriented design pattern variants, as well as any ad-hoc pattern (e.g., combination patterns) given their UML Class diagram. Our second contribution are insights to improving design pattern detection accuracy, such as incorporating behavioral aspects of a pattern in addition to structural characteristics by incorporating stereotypes, filters, and Sequence diagrams when necessary. Our final contribution is a repository of SPARQL queries that contains object-oriented design patterns and their variants [11].

We assessed PatternScout using experiments to measure accuracy. Our results indicate that they are comparable, or outperform existing techniques (e.g., [4, 12]).

2 Detecting Design Patterns

In this section, we discuss key concepts for automatically generating queries from UML Diagrams.

DOI reference number: 10.18293/SEKE2022-167

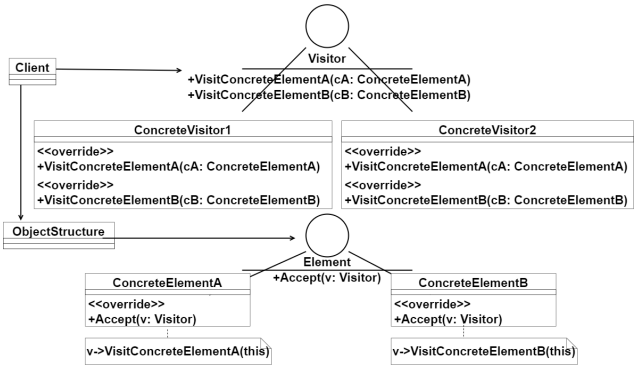


Figure 1: UML Representation of the Visitor Pattern

2.1 Identify Pattern Characteristics

Design pattern characteristics can be extracted from both UML Class and Sequence Diagrams. A Class diagram shows the objects within a design pattern and static relationships between those objects, while Sequence diagrams shows interactions between objects. We discuss how we use these diagrams in detecting patterns.

In a UML Class Diagram, PatternScout extracts relationships between classes, between classes and methods, between classes and attributes, and between methods and parameters, such as **Contain Relationships** (hasMethod, hasType, hasReturnType, hasModifiers, hasField, hasParameter, hasConstructor) and **Class Relationships** (Association, Generalization, Aggregation, Composition, Interface Realization, Dependency). PatternScout also extracts the following: **OO Entities** (Classes, Methods, Constructors, Fields, Method Parameters, Interfaces), **Visibility/Property** (Public, Private, Protected, Static, Final, Synchronized, Abstract), **Stereotypes**(Constructor, Override), and **Interactions**(Method Invocations).

We generate a SPARQL query by including relevant entities (i.e., "OO Entities") of the design pattern in the SELECT clause. We then add characteristics in the WHERE clause. For example, a project that contains a visitor pattern may contain the following RDF triples in its RDF graph:

```
PREFIX woc: <http://rdf.webofcode.org/woc/>
woc:SoldierVisitor woc:implements woc:UnitVisitor .
woc:SoldierVisitor woc:hasMethod woc:SoldierVisitor-vistSoldier() .
woc:SoldierVisitor-visitComander woc:hasParameter
woc:visitComander(com.iluwater.visitor.Commander)-parameter-0 .
woc:SoldierVisitor-visitCommander(com.iluwater.visitor.Commander)
-parameter-0 woc:hasType woc:Commander  .
```

These characteristics are captured in a UML Class diagram of a Visitor pattern as shown in Figure 1. To generate a SPARQL query, we extract all the entities in the Class diagram, such as class names and method names and add them to the SELECT clause, as shown below.

```
SELECT ?Visitor27 ?VisitConcreteElementA11 ?VisitConcreteElementB13
?VisitConcreteElementA27 ?VisitConcreteElementB29 ?Accept13
?Accept16 ?Accept20 ?VisitConcreteElementA24
?VisitConcreteElementB26 ?ConcreteVisitor15 ?Element14
?ConcreteVisitor211 ?ConcreteElementA18 ?ConcreteElementB22
```

We also add triples defining the type (role) of each entity to the WHERE clause. For example, if a Method is encountered, woc:Method type is added for that component in the WHERE clause:

```
SELECT ?ClassA ?OperationA
WHERE {
?ClassA  a woc:Class .
?OperationA  a woc:Method .
```

Next, we add the structure of the pattern, such as the aforementioned relationships between entities to the WHERE clause. A relationship is represented by an RDF triple in the format *(fromItem, relationshipType, toItem)* representing a relation from *fromItem* to *toItem*. Both *fromItem* and *toItem* are OO Entities. In the following snippet, ClassA is the *fromItem*, OperationA is the *toItem* and woc:hasMethod is the *relationshipType*.

```
?ClassA woc:hasMethod ?OperationA .
```

If characteristics related to visibility, property, data types, and return types are included in a Class diagram, PatternScout also generates the corresponding triples. Each line is a condition. All conditions in a WHERE clause must be satisfied for a design pattern match to occur. The greater the conditions, the more specific and restrictive the queries become. On the other hand, fewer conditions provide more allowance for variation in implementation. A partial list of characteristics for the above Class Diagram that would be included in a WHERE clause is below:

```
?Visitor27  a woc:Interface .
?ConcreteVisitor211  a woc:Class .
?ConcreteVisitor211 woc:implements ?Visitor27 .
?ConcreteVisitor211 woc:hasMethod ?VisitConcreteElementA11 .
?VisitConcreteElementA11 woc:hasParameter ?cA10 .
?cA10 woc:hasType ?ConcreteElementA18 .
```

Some design patterns such as the Visitor pattern require behavioral information to accurately identify it. Behavioral specifications related to method invocation can be obtained from Sequence diagrams. Here is an example snippet from [13] that shows an RDF triple with a behavioral characteristic of Visitor design pattern pattern.

```
<http://rdf.webofcode.org/woc/com.iluwatar.visitor.Sergeant-
accept(com.iluwatar.visitor.UnitVisitor)>
<http://rdf.webofcode.org/woc/references>
<http://rdf.webofcode.org/woc/com.iluwatar.visitor.UnitVisitor-
visitSergeant(com.iluwatar.visitor.Sergeant)> .
```

A SPARQL query based only on the Class diagram of a Visitor pattern will include the following triple representing an association relationship:

```
?ConcreteElementA18    woc:references    ?Visitor27    .
```

This structural relationship will result in false positive results as it does not describe the defining characteristics of a Visitor pattern. Moreover, the RDF graph representation

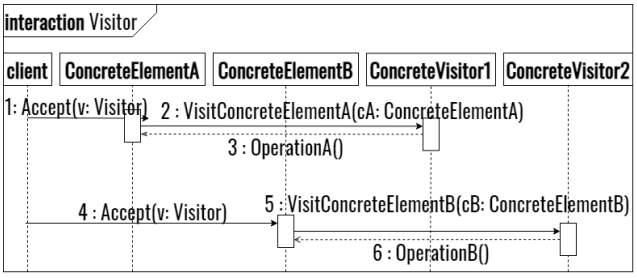


Figure 2: Sequence Diagram for Visitor Design Pattern

of the code might not include the relevant triple with references relationship if the two classes are under the same package, causing false negatives. However, by including information from the Sequence diagram shown in Figure 2, the WHERE statement would include the invocation of VisitConcreteElement method in the Accept method. This not only reduces false positives with a defining characteristic of the pattern, but is also immune to false negatives as the triple will be part of the RDF graph irrespective of the code base structure. The RDF triple is as follows:

```
?Accept16    woc:references  ?VisitConcreteElementA24  .
```

By including Sequence diagrams and consequently method invocation characteristics, PatternScout can distinguish between otherwise structurally identical design patterns (e.g., State - Strategy, Adapter - Command).

2.2 Use Stereotypes

Another way to improve accuracy of design pattern identification is to use stereotypes. Although not part of the standard UML specification, stereotypes have been used to differentiate or represent features like Constructors, Getters, Setters and Overriding of methods. Using stereotypes, Constructors can be differentiated from other Methods using *woc:Constructor* instead of the *woc:Method* type and *woc:hasConstructor* instead of the *woc:hasMethod* relationship. Similarly, methods of a child class that overrides methods of a parent class are differentiated with *woc:hasAnnotation overrides* relationship. We observed this to significantly reduce false positives, especially in detecting patterns like Proxy, Builder and Singleton.

2.3 Accommodate Variations with Query Map

Design patterns not only have many implementation variations, but some variations are combinations of existing patterns (e.g, Visitor Combinator patterns is a variant of the Visitor pattern where the GoF specification of Visitor is combined with Composite pattern for object oriented tree traversal). PatternScout can handle these variations as long as they can be represented as a Class diagram.

We use a query map to efficiently connect variants with each pattern. Instead of running one SPARQL query at a time (as shown in Figure 3), we run multiple queries at once. Each design pattern has its own section. Below each design pattern is a list of variants, with the variant name and filename as shown below:

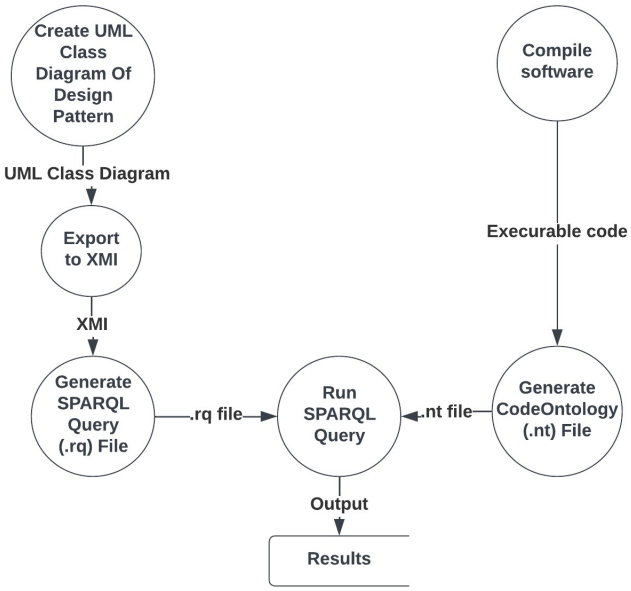


Figure 3: Approach Overview

```
"Visitor": {
    "Visitor GoF": "visitor.rq",
    "Visitor Combinators": "visitor_combinators.rq"
},
"Singleton" : {
    "Singleton GoF" : "singleton.rq",
    "Eager Instantiation": "singleton_Eager_Instantiation.rq",
```

3 Validation

We conducted two analyses to evaluate if the SPARQL queries generated by PatternScout are accurate and sufficient for detecting design patterns. The approach is summarized in Figure 3. The pre-processing required is as follows: generating an RDF graph for each project, creating a Class diagram in an XMI format and generating SPARQL queries from the Class diagram. Creating Class diagrams and generating SPARQL queries is a one time task for each pattern variant and can be reused for any project. The repository of SPARQL queries for known variants of object oriented design patterns are packaged with PatternScout. The only project-specific pre-processing needed is generating an RDF graph, using CodeOntology [14]. The preparation time taken for each project is shown in Table 1.

3.1 Detecting Presence/Absence of Patterns

Experiment: We ran an experiment to determine if PatternScout can correctly detect the presence/absence of patterns. Since these pattern instances in the selected projects are widely studied in literature (e.g., [15, 16, 12]), we used their results as our gold standard for this analysis. If a pat-

Open Source Project	LOC	Java Classes	RDF triples	Preparation Time (ms)
JHotDraw v5.1 (JHD)	8907	155	52824	2040
JRefactory v2.6.24 (JRF)	56187	569	70178	3023
JUnit v3.7 (JUN)	1347	33	9497	2001
QuickUML 2001 (QUM)	9250	156	59480	1756
MapperXML 1.9.7 (MPX)	14928	217	17147	6002
Dom4J v1.6.1 (DOM)	26350	328	29874	2059
Lizzy v1.1.1 (LZ)	12915	197	11617	1083

Table 1: Projects used for evaluation

	JHD	JRF	QUM	JUN	DOM	MPX	LZ	P	R
FM*	✓ | ✓	✓ | ✓	× | ✓	× | ✓	✓ | ✓	✓ | ×	✓ | ✓	67.67	80
PRTT	✓ | ✓	× | ×	✓ | ×	× | ×	✓ | ×	× | ×	× | ×	100	33.33
SGLT	✓ | ✓	✓ | ✓	✓ | ✓	× | ×	✓ | ✓	✓ | ✓	✓ | ✓	100	100
TPLT	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ✓	100	100
STT	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ×	× | ✓	✓ | ✓	83.33	83.33
CMD	✓ | ✓	× | ×	✓ | ×	× | ×	× | ×	× | ×	× | ×	100	50
OBSV	✓ | ✓	✓ | ✓	✓ | ✓	✓ | ×	✓ | ×	✓ | ✓	× | ×	100	66.67

Table 2: P&R based on presence(✓) or absence(×) of Patterns. Tuple represent (Actual Label | Predicted Label)

*FM: Factory Method, PRTT: Prototype, SGLT: Singleton, TPLT: Template Method, STT: State, CMD: Command, OBSV: Observer

tern is reported in a project (regardless of variation, as previous studies did not specify the variant used), we used a check mark (or True) for the actual label. If it is reported as absent, we used an X-mark (or False) (See Table 2).

Result: We summarize the precision and recall in detecting each pattern in the P and R columns of Table 2. Based on these 7 projects, we get an average precision of 92.86% and average recall of 73.33%. During manual inspection, we observed that the low recall was often due to SPARQL not interpreting the transitive nature of the inheritance relationship when parsing triples in the RDF graph.

3.2 Comparing PatternScout with Existing Tools

Experiment: We also ran experiments to compare the accuracy of PatternScout with existing tools (Table 3). In order to calculate precision and recall of each tool, we identified the true instances of each pattern in the projects used for benchmarking. We established the ground truth through manual inspection and validated with other results [15, 17]. We compared unique instances of patterns retrieved by PatternScout with Finder [18] and DPD [12]. While SparT [17] was also evaluated, we excluded it from the analysis as the off-the-shelf implementation did not include specifications for creational and structural patterns. We excluded Tools that are unavailable for download (e.g., [19, 20]) or those for which a core dependency is deprecated (e.g., [21, 15]) from the comparison.

We executed the benchmarked tools, DPD and SparT, on a Windows 10 21H2 Virtual Machine. FINDER was executed on a Rocky Linux 8.5 Virtual Machine. Java arguments and runtime parameters for execution were as recommended by the tools. SparT did not utilize Java, but ran natively on Windows.

	PatternScout			DPD			FINDER		
	P	R	f1	P	R	f1	P	R	f1
FM	68.75	100	77.27	25	50	66.67	20	50	57.14
PRTT	100	33.33	50	100	100	100	100	33.33	50
CMD	88.89	57.14	69.57	100	57.14	72.73	55	78.57	64.71
STT	41.67	100	58.33	14.94	79.41	23.06	25	50	66.67
SGLT	100	100	100	100	100	100	66.67	55.56	90
TPLT	88.89	87.78	87.88	83.33	74.44	78.45	72.22	80	74.81

Table 3: Detection accuracy on JHD, MPX and QUM

Results: Precision, Recall and F1-score were calculated on JHD, MPX and QUM as these systems were included in the required input formats with the distributions of the selected tools (see Table 3). The average precision, recall and F1-score of SPARQL queries generated with PatternScout(81.37%, 79.71%, 73.84%) are better than DPD(70.55%, 76.83%, 73.49%) and FINDER(56.48%, 57.91%, 67.22%). DPD and FINDER rely exclusively on code structure whereas queries generated by PatternScout are able to detect both code structure and behavior.

4 Discussion: Precision & Recall Tradeoff

We now cover threats to validity, the language-agnostic potential of our tool as well its present limitations. The tradeoff in precision and recall depends upon how restrictive the SPARQL query is. For the same design pattern, the SPARQL query can be more restrictive if there are conditions specifying visibility, property, stereotypes, etc, and less restrictive if these constraints are relaxed. While a more restrictive query can reduce false positive results (increase precision), this can fail to retrieve some instances that do not conform strictly to the structure of a pattern. For patterns like Singleton where access modifiers of the constructor and instance are important, a more restrictive query is appropriate. To achieve a balance between precision and recall, we can use a query map to identify variants to detect. The tolerance for false positives and false negatives vary for different usecases (e.g., detecting SDPs to ensure a security concern is addressed needs higher precision over recall).

5 Related Work

Earlier approaches for detecting design patterns in source code ranged from sub-graph matching [22, 23] and ontology based techniques [4] to using machine learning (ML) techniques [2, 12]. A detailed meta-analysis of various design pattern mining approaches is discussed in [24]. Summary of comparison is in Table 4.

6 Conclusion

PatternScout is a lightweight tool that automatically generates SPARQL queries from Class and Sequence diagrams to find design patterns in source code. The generated query has the same granularity as input diagrams in terms of entities and relationships between those entities. Thus, it is able to identify more types of patterns than other techniques.

Technique	Limitation	PatternScout
Semantic Web / Ontology [4]	Supports variants with same number of targets; requires manual creation of queries/rules for each pattern	requires presence of Class/Sequence diagrams, many of which already exist
ML / Code Metrics [2] [25]	accommodates variations but compromises accuracy; requires manual training for each pattern	accommodates variations without compromising accuracy; no manual training
Subgraph Matching [23] [26]	can't capture design- and code-level concepts; subject to false negatives as results are very specific	captures design- and code-level concepts; minimizes false negatives using query maps

Table 4: PatternScout with Existing Techniques

While we primarily focused on OO design patterns, SDPs that can be expressed as Class or Sequence diagrams can also be detected. We evaluated PatternScout using representative patterns from three types of design patterns: creational, structural, and behavioral. Precision and recall on the open source projects indicate that our technique is comparable to, or better than related tools.

Acknowledgment

The authors thank Elif Hepateskan, Conor Barrett, Sonam Misra, Aashima Mehta, Namita Dave, Aidar Kurmanbek-Uulu, Zhijun Huang, and Logan Petersen for their assistance with performing evaluations. Matthew Hewitt assisted with evaluation and related work. Jacob McHugh assisted with providing Query Map feature. This effort is supported in part by the University of Washington Bothell Computing and Software Systems (CSS) Division Project & the CSS Division Graduate Research funds.

References

[1] M. VanHilst and E. B. Fernandez, "Reverse engr to detect security patterns in code," in *Proc. Int'l Workshop on Software Patterns & Quality. Info Processing*, 2007.

[2] S. Uchiyama, A. Kubo, H. Washizaki, Y. Fukazawa, and others, "Detecting design patterns in object-oriented program source code by using metrics & machine learning," *Journal of Software Engr & Applications*, vol. 7, no. 12, 2014.

[3] J. Niere, M. Meyer, and L. Wendehals, "User-driven adaption in rule-based pattern recognition," University of Paderborn, Germany, Tech. Rep. tr-ri-04-249, 2004.

[4] S. Paydar and M. Kahani, "A semantic web based approach for design pattern detection from source code," in *Proc Int'l eConf on Computer & Knowledge Engr*, 2012.

[5] M. Bunke, "Security-Pattern Recognition & Validation," PhD Thesis, Universität Bremen, 2019.

[6] J. Potoniec, "Learning SPARQL Queries from Expected Results," *Computing & Informatics*, vol. 38, no. 3, 2019.

[7] F. Haag, S. Lohmann, and T. Ertl, "SparqlFilterFlow: SPARQL Query Composition for Everyone," in *The Semantic Web: ESWC Satellite Events*, 2014.

[8] A. Ampatzoglou, S. Charalampidou, and I. Stamelos, "Research state of the art on GoF design patterns: A mapping study," *Journal of Systems & Software (JSS)*, vol. 86, no. 7, 2013.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.

[10] C. Dougherty, K. Sayre, R. C. Seacord, D. Svoboda, and K. Togashi, "Secure design patterns," CMU Softw Engr Inst, Tech. Rep., 2009.

[11] G. Rasool and H. Akhtar, "Towards A Catalog of Design Patterns Variants," in *Int'l Conf on Frontiers of Info Tech*, 2019.

[12] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis, "Design pattern detection using similarity scoring," *Trans on Software Engr (TSE)*, vol. 32, no. 11, 2006.

[13] "Design patterns implemented in java," https://github.com/iluwatar/java-design-patterns/l, (accessed: 05.27.2020).

[14] M. Atzeni and M. Atzori, "CodeOntology: RDF-ization of source code," in *Int'l Semantic Web Conf*. Springer, 2017.

[15] A. D. Lucia, V. Deufemia, C. Gravino, and M. Risi, "Detecting the behavior of design patterns through model checking & dynamic analysis," *Trans on Softw Engr & Methodology*, vol. 26, no. 4, 2018.

[16] D. Yu, Y. Zhang, and Z. Chen, "A comprehensive approach to the recovery of design pattern instances based on sub-patterns & method signatures," *Journal of Systems & Software*, vol. 103, 2015.

[17] R. Xiong, D. Lo, and B. Li, "Distinguishing Similar Design Pattern Instances through Temporal Behavior Analysis," in *Proc Int'l Conf on Softw Analysis, Evolution & Reengineering*, 2020.

[18] H. Dabain, A. Manzer, and V. Tzerpos, "Design pattern detection using FINDER," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 1586–1593.

[19] M. L. Bernardi, M. Cimitile, and G. Di Lucca, "Design pattern detection using a DSL-driven graph matching approach," *Journal of Software: Evolution & Process*, vol. 26, no. 12, 2014.

[20] G. Rasool and P. Mäder, "A customizable approach to design patterns recognition based on feature types," *Arabian Journal for Science & Engr*, vol. 39, no. 12, 2014.

[21] A. Binun and G. Kniesel, "Joining forces for higher precision and recall of design pattern detection," *CS Department III, Uni. Bonn, Germany, Technical report IAI-TR-2012-01*, 2012.

[22] D. Yu, Y. Zhang, J. Ge, and W. Wu, "From sub-patterns to patterns: an approach to the detection of structural design pattern instances by subgraph mining & merging," in *Proc Comp Softw & App Conf*, 2013.

[23] M. Gupta and A. Pande, "Design patterns mining using subgraph isomorphism: Relational view," *Int'l Journal of Softw Engr and Its App*, vol. 270.

[24] J. Dong, Y. Zhao, and T. Peng, "A review of design pattern mining techniques," *Int'l Journal of Software Engr & Knowledge Engr*, vol. 19, no. 06, 2009, publisher: World Scientific.

[25] F. Tie, J. Le, Z. Jiachen, and W. Hongyuan, "Design pattern detection method based on stacking generalization," *Journal of Software*, vol. 31, no. 6, 2020.

[26] W. Liu, C. Zhang, F. Wang, and Y. Yang, "Combining Network Analysis with Structural Matching for Design Pattern Detection," in *Proc Evaluation & Assessment in Softw Engr*, 2020.

A Distributed Graph Inference Computation Framework Based on Graph Neural Network Model

Zeting Pan, Yue Yu, Junsheng Chang*
College of Computer
National University of Defense Technology
Changsha, China
{pannudt, yuyue, junshengchang}@nudt.edu.cn

***Abstract*—A graph is a structure that can effectively represent objects and the relationships between them. Graph Neural Networks (GNNs) enable deep learning to be applied in the graph domain. However, most GNN models are trained offline and cannot be directly used in real-time monitoring scenarios. In addition, due to the very large data scale of the graph, a single machine cannot meet the demand, and there is a performance bottleneck. Therefore, we propose a distributed graph neural network inference computing framework, which can be applied to GNN models in the form of Encoder-Decoder. We propose the idea of "single-point inference, message passing, distributed computing", which enables the system to use offline-trained GNNs for real-time inference computations on graph data. To maintain the model effect, we add the second-degree subgraph and mailbox mechanism to the continuous iterative calculation. Finally, our results on public datasets show that this method greatly improves the upper limit of inference computation and has better timeliness. And it maintains a good model effect on three types of classical tasks. The source code is published in a Github repository.**

Keywords-component; graph inference; graph neural network; distributed graph computing

I. INTRODUCTION

A graph is an abstract data structure. A graph G = (V, E) consists of a vertex set V and an edge set E, which can be used to represent multiple objects and the relationship between them. Initially, scholars' research on graphs mainly focused on static graphs, that is, without considering temporal information. With the increase in applications, static graphs can no longer meet practical requirements, so more researchers begin to explore dynamic graphs, from discrete-time dynamic graphs to continuous-time dynamic graphs [1]. Generally speaking, for a dynamic graph, the vertices on the graph be represented as $\forall\ v_i \in V$, v_i=(id, feat, timestamp), i=1,2,..., and the edges bet can be represented as $\forall\ e_i \in E$, e_i=(src, dst, feat, timestamp), i=1,2,.... These properties can be summarized as identifiers, characteristics, and timestamps.

In practical applications, the scale of graph data is often very large, such as payment transactions, social interactions, and biological information [2]. A survey [3] showed that the graphs in practice typically contain more than 1 billion edges. Another survey [4] noted that over 68.5% of tasks applied machine learning algorithms (clustering, regression, etc.) on the graph. The popularity of deep learning has also prompted people to migrate it to the graph domain, such as graph neural networks (GNNs). The more well-known networks in GNNs are GCN [5], GAT [6], TGAT [7], etc. They mainly perform three types of tasks: link prediction (LP), node classification (NC), and edge classification (EC). And it has a good effect on task accuracy [8].

However, many problems arise when applying GNN models to graph inference computations. The first is the single-machine capacity problem. A single machine cannot withstand large-scale graph data, and there will be serious performance bottlenecks. Therefore, a distributed computing environment has become an urgent need for graph computing. Apache Spark [9] is a cluster computing framework, and GraphX [10] is a distributed graph processing framework. GraphX can be used to express graph computations, but it does not directly support GNNs. The second is the real-time problem of graph computing. The offline training method of GNNs limits its application to scenarios with low real-time requirements. If applied to payment security, the application will not be able to quickly intercept fraud, money laundering, and other dangerous behaviors. In conclusion, it is important to apply offline-trained GNN models to distributed environments for real-time inference computation. Therefore, this paper proposes a framework that maintains GraphX features while supporting GNN models, enabling real-time distributed graph inference computation.

The framework proposed in this paper is mainly to solve the following two problems. (1) How to apply the GNN model to a distributed system for graph inference computation. In the application, the GNN model does not support serialization and cannot meet the distributed requirements. To solve this problem, we first modify the model input and output so that they can be directly used for online inference. Next, we adopt the idea of "single-point inference, message passing, distributed computing". That is, the model is stored in a distributed manner, and models on different machines are dynamically called when used. Finally, the output of the model is passed as a message to the relevant vertices. In this way, the passed message size is reduced from model level (MB-GB) to matrix level (KB-MB). (2) How to perform real-time inference calculations and maintain model performance. When an event occurs, the scope of influence is often more than the source vertex (src) and the destination vertex (dst). Therefore, we have iteratively updated graph properties through incremental composition, computing

*Corresponding author.
DOI reference number: 10.18293/SEKE2022-042

second-degree subgraphs, and mailbox mechanisms. These steps will be disassembled into many tasks during execution and distributed to multiple worker nodes for parallel computing. In this way, we can apply the GNN model in the form of Encoder-Decoder to the actual environment for real-time distributed graph inference calculations. The contributions of this paper are summarized as follows:

- We put forward the idea of "single-point inference, message passing, distributed computing" so that the GNN model in the form of Encoder-Decoder can be applied to a distributed environment.
- We propose a method based on incremental composition, constructing second-degree subgraphs, and maintaining mailboxes so that distributed inference computing can ensure both timeliness and effects.
- Finally, we implemented the framework and tested it on Wikipedia and Reddit. The results show that the single-event inference time for a thousand events is 1.6203s, which is only 36.19% longer than that of a single machine under the same conditions, but the throughput is improved by 116.89%. In addition, the effect of three categories of tasks is maintained, among which the accuracy of the Wikipedia LP task is 87.03%.

II. Related Work

A. Distributed graph computing framework

MapReduce is a simple distributed computing framework that facilitates the processing of massive graph data, but it cannot iteratively compute efficiently. Bulk synchronous parallel (BSP) [11] proposed by Valiant in 1990 is suitable for iterative computing of graphs, which decomposes tasks into a series of iterative operations. Inspired by BSP, Google proposed the first vertex-centric distributed graph computing framework Pregel [12] in 2010. Since then, researchers have successively proposed a variety of distributed graph computing frameworks, including PowerGraph [13], GraphHP [14], and Hybrid [15]. These frameworks can efficiently perform graph iteration algorithms. However, their limited expressive computation makes it difficult to express important stages in a typical graph analysis pipeline, such as graph modification, cross-graph computation, etc.

Spark GraphX is a distributed graph processing framework, and its core abstraction is Resilient Distributed Property Graph, a directed multigraph with properties on both nodes and edges. GraphX extends the abstraction of Spark RDD and has two views (table and graph), and only needs one physical storage [16]. These two views have their unique operators, to obtain flexible operation and execution efficiency. In terms of calculation, all operations on the view will be converted into RDD operations of the associated table view to complete. In this way, graph computation is equivalent to the transformation process of a series of RDDs. Therefore, GraphX finally has three key features: Immutable, Distributed, and Fault-Tolerant.

B. Graph Neural Network

Dynamic graph representation has gone through four main stages of development, namely static, weighted edge, discrete, and continuous. There are also several types of methods for learning dynamic graph representations, including tensor decomposition, random walk, and deep learning. Among them, GNNs in deep learning methods have received extensive attention because they can combine time series encoding with aggregation of adjacent nodes. GCN [5] learns features better by aggregating the information of neighbor points, but it cannot use temporal information. However, in the field of graphs, timing has an important influence on the change of vertex-edge relationship, so the research direction of GNNs gradually shifts from static to dynamic.

Dynamic graph neural network (DGNN) aggregates deep time series encoding and node features and is mainly divided into discrete and continuous categories. Discrete DGNN first uses a certain GNN to obtain vertex embedding and then uses a certain RNN or Attention network for time series modeling. The representative networks are DySAT [17], etc. Most of the current methods of continuous DGNN use snapshot modeling, which is only a rough estimation of time, and related networks include TGAT [7], TGN [18], etc. TGN attempts to introduce the Encoder-Decoder neural network framework in the graph field. The encoder is responsible for encoding the vertex and edge features on the graph into vectors, and the decoder calculates and predicts attribute values for the encoded vectors according to specific executing tasks. This form of decoupling enables real-time inference computations on graphs.

III. Method

This section will first introduce the overall framework and then introduce the three main modules in the framework in detail.

A. Overall Framework

Fig. 1 presents an overview of our distributed graph inference. It is a distributed graph inference framework that supports the Encoder-Decoder form of GNN. The framework is mainly composed of the following modules: incremental composition, second-degree subgraph calculation, GNN encoder, mailbox, and GNN decoder.

- **Incremental composition module** corresponding to steps (a)-(b). When a new event is generated in the data source, the event will be added to the vertexRDD and edgeRDD of the historical graph event, and incremental composition will be performed to obtain the whole graph. Note that if it is an undirected graph, for an event e, we will generate both forward and reverse edges.
- **Second-degree subgraph calculation module**, which is the "Full Graph and 2D-Subgraph" in the figure. After the whole graph is obtained, the basic properties of some vertices and edges will be updated according to e and its second-degree subgraph will be calculated.
- **GNN encoder module** corresponding to steps (c)-(e). After loading the trained GNN encoder model, the second-degree subgraph feature matrix is used as the input of the model. Then, the embedding of each vertex in the second-degree subgraph can be obtained and then updated to the whole graph.

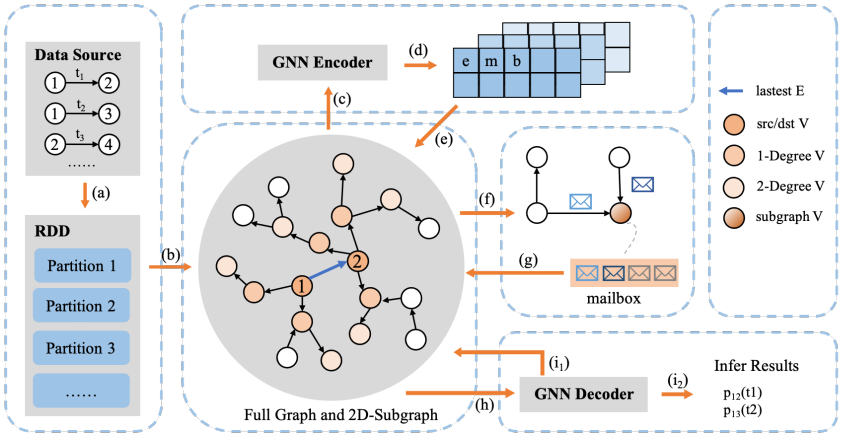


Figure 1. Distributed graph inference system framework.

- **The mailbox module** corresponds to steps (f)-(g). A mail will be sent along each edge in the second-degree subgraph, and the main content of the mail is the current features of the edge and some historical interaction information. The vertex that receives the mail will add it to its mailbox.
- **GNN decoder module** corresponding to steps (h)-(i). The above results will be decoded, and logical inference results will be given according to the types of tasks performed (LP/NC/EC).

B. *Second Degree Subgraph Algorithm*

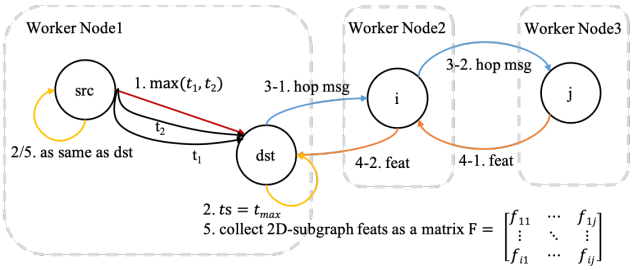


Figure 2. Second-degree subgraph algorithm.

When a new event e=(src, dst, timestamp, feat) occurs, the edges with the same starting vertex and destination vertex are first merged, as shown in Fig. 2. The way to merge is to leave the edge with the largest timestamp. Since historical events are often embedded into features by previous inferences, merging duplicate edges does not affect results. It can also reduce the number of edges in the graph to save resources, especially when the vertex-to-edge ratio is large. Then, as shown in step 2 in Fig. 2, the timestamps of src and dst will be updated to $\max(t_1, t_2)$.

Due to the particularity of graphs, an event often affects more than src and dst . Considering the effect and performance comprehensively, we decided to control its influence range within the subgraph reachable by two hops. To obtain this subgraph, we design steps 3 and 4, namely the sending of hop messages and the return of feature messages. From src and dst, two rounds of message sending operations will be performed. The first round will send its hop-1 to the neighbor and update the hop value of the neighbor vertex. In the second round, the neighbor vertex sends its hop-1 to its neighbor (excluding src and dst) and updates the hop value of the vertex that receives the message. This obtains a second-degree subgraph about the new event e. Step 4 is the opposite of Step 3, which will transmit its feature information back from the second-degree vertex. After two rounds of message return, src and dst will aggregate the features of all vertices in the second-degree subgraph. Step 5 stores these features in src as one of the attributes of the vertex.

C. *GNN Encoder algorithm*

Trained GNN models tend to take up a lot of storage space and are difficult to serialize. Even with serialization methods, transferring models between machines incurs a significant bandwidth overhead, which can severely impact performance. Therefore, the inference algorithm we designed will use the spark driver to uniformly manage the scheduling of machines, so that each worker node has a copy of the GNN model, as shown in Fig. 3. The model can be loaded directly and dynamically during the inference process, without the need for network transmission. Furthermore, all vertices in a second-degree subgraph need to be inferred, but doing inference at the same time introduces additional overhead. Therefore, we propose an algorithm for "single-point inference, message passing, distributed computing". Based on the subgraph collected in the previous step, we only load the model in src or dst, then get the model output and update the vertex features. Finally, the model output is sent to other vertices in the subgraph in the form of a message to complete the vertex update. For the GNN decoder algorithm, the process is the same as the GNN encoder. Due to space limitations, we will not repeat them here.

D. *Mailbox algorithm*

As shown in Fig. 3, after updating the vertex features using embedding, we will generate a mail along each edge in the subgraph (mail = feat_{src} + feat_{dst} + feat_{e}), and then sent it to the destination vertex. Each vertex in the subgraph will take an average of all the received emails, and then add it to the mailbox. That is to say, a vertex can only have one mail added to its mailbox in one iteration, which solves the problem of supernode message explosion. The mailbox maintained by each vertex is implemented using a list, and the default maximum capacity of the list is 10 (this value can be modified more practically). When

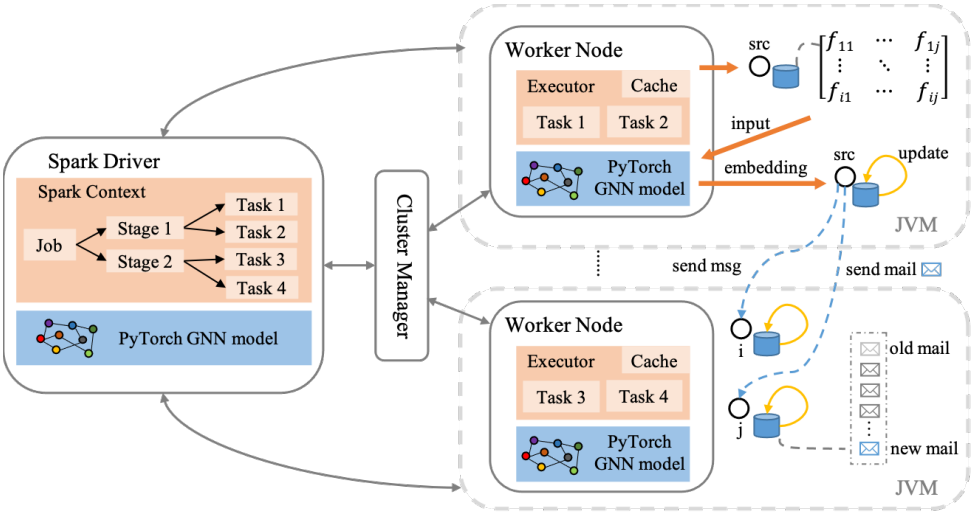


Figure 3. The execution process of inference and computation.

there are less than 10 mails, new mail will be added directly to the end of the list. When it is greater than 10, the oldest mail in the list will be deleted to leave space for new mail. Through the mechanism of the mailbox, the storage of each vertex itself includes vertex features, information about neighbor vertices, and features brought by historical events. When inferring, it can be encoded into the form of a matrix and directly input into the model.

E. Algorithm Pseudocode

Through the above algorithm, we realize distributed graph inference computing. That is, graphs can be incrementally composed and iteratively performed graph computation, graph inference, and graph update. The pseudocode of the overall logic is shown in Table 1 below.

TABLE I. PSEUDOCODE FOR DISTRIBUTED GRAPH COMPUTING INFERENCE

Algorithm: Distributed graph computing inference
Input: event data source e=(src,dst,feat,timestamp)
Output: logical inference results and accuracy
initialize spark and static resources config
use n-events warm up inference
while has new event e **do**
if $v_{src/dst} \notin$ vRDD **then**
initialize $v_{src/dst}$ and add them to vRDD
add e to eRDD, then create graph with vRDD and eRDD
merge duplicate edges and update $v_{src/dst}$ with the timestamp of the lastest edge
for v_{src} or v_{dst} **do**
send hop_{v_i}-1 to its neighbors
for each vertex that receives src or dst hop messages **do**
send hop_{v_j}-1 to its neighbors
collect all vertex features of received hop messages to src as 2D-subgraph
update feat of subgraph vertices to embedding=Encoder(2D-subgraph)
for each $e_i \in$2D-subgraph **do**
send mail messages to the dst along the edge, and calculate the average value of all mails at dst
for each $v_i \in$2D-subgraph **do**
if len($mailbox_i$)≥10 **then**
remove the oldest mail from $mailbox_i$ header
add average mail to the tail of the $mailbox_i$
for triplets(src,dst,e) **do**
update the logical results of triplets to logit=Decoder(concat(src,dst,e))
calculate the inference results of the whole graph to get the accuracy
return accuracy

IV. EXPERIMENTS

In this section, we will first introduce the experimental setup, including the device environment, datasets, GNN models, and inference system. Next, the results of the experiment will be explained in terms of performance and effects. The source code of our system is published at a Github repository[1].

A. Setting

- **Hardware and software environment.** Due to the limitation of cluster resources, this experiment uses multithreading to simulate a distributed environment. The processor is Intel(R) Core(TM) i5-8257U CPU @2.00GHz, 16GB memory, 500G hard disk. The operating system is macOS Catalina 10.15.7, and the development tool is IntelliJ IDEA 2020.1.2. The running environment is Spark 3.2.0, Hadoop 3.3.0, Scala 2.12.15, and Java 1.8.0.
- **Graph datasets.** This experiment uses Wikipedia [19], and Reddit [19] public datasets for experiments. Among them, Wikipedia represents the interaction between users and wiki pages. Reddit represents the interaction events of users in social networking. The timestamps of all edges in both datasets span 30 days.
- **GNN models.** The framework proposed in this paper is suitable for GNNs in the form of encoder-decoder, we chose APAN [20] and reproduced it. Considering that Java does not support graph input when calling the model, we modified the input and output data form of APAN to matrices. The main parameters when training the model are: the maximum epoch is 50, the batchsize is 100, the initial learning rate is 0.0001, and the dropout

[1]https://github.com/napdada/Distributed-Graph-Inference-Java

TABLE II. GRAPH INFERENCE CALCULATION RESULTS AND MAIN STEPS TIME-CONSUMING IN TWO ENVIRONMENTS

Environment	LP task inference time(s)		Throughput cap	Time-consuming inference calculation of specific steps(ms)								
	1000 events	*1000 events(*)*		*Create Graph*	*Merge Edges*	*Update Ts*	*2D-Subgraph*	*Encoder*	*Send Emb*	*Mailbox*	*Decoder*	*Evaluate*
SM	1.1897	1.1210	1835	4.24	0.37	1.76	733.51	2.16	374.62	3.62	0.44	68.79
DM	1.6203	1.5335	3980	4.40	0.38	1.84	951.76	2.24	568.35	3.77	0.48	86.83

a. SM: single machine environment, DM: distributed machine environment. The names of the specific steps from left to right represent incremental composition, merge duplicate edges, update timestamp, generate second-degree subgraph, call encoder model, send embedding, transfer mail, call decoder, and evaluate results

is 0.1. The ratio of training, testing, and validation data is 7:1.5:1.5. After training, the ".pt" models of three types of tasks (LP/NC/EC) on two datasets are obtained, and the effect is the same as in the original paper.

- **Inference system.** Considering the system startup process and graph initial features are zero, we set a warm-up process of inference. That is, by default, the first ten inferences are not included in the result. The number of distributed cores and partitions is both 2, and the partition strategy is EdgePartition2D. The JVM parameters will be adjusted according to the executed tasks, mainly adjusting the memory size. In addition, to reduce the effect of chance, all results are the mean of ten replicate experiments.

B. *Inference performance*

1) *Timeliness and throughput caps.*

We reproduce APAN and apply it to graph inference computation to compare single machine and distributed. We configure 4G memory for a single machine. We configured 2-core 8G memory for the distributed environment, which is equivalent to two single machines working at the same time. After that, we calculated the average time and upper limit of these two environments as shown in Table 2. It can be seen that when executing the LP task with a thousand events on Wikipedia, the distributed time is longer than that of a single machine. However, the upper limit of distributed computing is increased, and the number of iterations in a single-machine environment will overflow when about 1800. That's to say, in the face of large graph data scenarios, distributed inference computing can solve the capacity bottleneck problem of a single machine.

To further explore the time-consuming, we have detailed statistics on the time-consuming of the main steps, as shown in Table 2. The data in the table visually show that the extra time is mainly spent on 2D-Subgraph, SendEmb, and Evaluate. Because in a single-machine environment, vertices and edges are stored on one machine, they only need to be read when they are used. However, vertices are stored in partitions according to the policies in a distributed environment, so data exchange in different partitions will bring additional communication overhead. Furthermore, to evaluate the results, the system needs to retrieve data from all partitions. This operation is time-consuming, and we count it into the results (the data marked with "*" in Table 2 are not included in the time-consuming statistics of this operation), so we believe that a single time consumption of 1.62s is reasonable. Comparing the results in the two environments, it can be seen that when the iterative inference is executed 1000 times, the distributed graph inference calculation uses 36.19% of the time overhead, in exchange for 116.89% of the larger throughput.

Considering that the graph continues to grow during the iterative process, if the resource consumption increases exponentially as the graph grows, the huge overhead will inevitably make the system worthless. To this end, we conducted inference experiments with varying numbers of events, ranging from 50 to 1000, within a range where problems such as overflow do not occur. The average time-consuming inference calculation is shown in Fig. 4. It can be seen that with the increase of the graph size, the inference computation time increases linearly. That is, our framework can be extended to work on clusters. In the face of large graph data, it can still guarantee the linear growth of resource consumption, rather than the problem of exponential explosion.

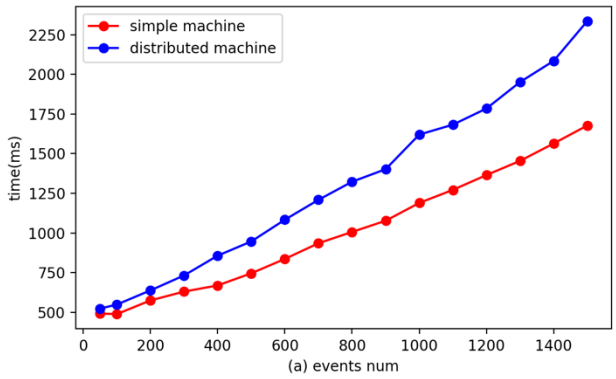


Figure 4. The relationship between the number of inference events and the time-consuming when performing LP tasks on Wikipedia

2) *Partitioning and Strategy.*

The number and strategy of partitions in a distributed environment often significantly impact the results, so we experimented with them. We set up a 4-core 5G distributed environment that used 100 events in Wikipedia to perform LP tasks. The results are shown in Table 3. Regarding the number of partitions, we can find that the computation time of graph inference increases rapidly with the number of partitions. Therefore, we need to dynamically adjust some configurations according to the actual situation. For small graphs, the number of partitions can be reduced. But for large graphs, we can increase the number of partitions and add machines to reduce the time-consuming impact. For the partitioning strategy, we can see that the random partitioning effect is the worst, followed by the EdgePartition1D (partitioning based on src only). The best partition strategy is EdgePartition2D (partition by both src and dst). This is in line with our understanding of graphs, that for edges, both the source and destination vertices are important.

TABLE III. THE EFFECT OF THE NUMBER OF PARTITIONS AND PARTITION STRATEGY ON TIME CONSUMPTION

Number of partitions	Partitioning strategy	Time (ms)
1	EdgePartition2D	606.24
2	EdgePartition2D	614.87
4	EdgePartition2D	655.79
8	EdgePartition2D	786.46
4	RandomVertexCut	674.38
4	EdgePartition1D	670.95

C. *Inference performance*

Table 4 shows the effect of APAN original data, reproduced APAN model data (APAN-Re), classic GNN model, and our distributed graph inference algorithm. That is the accuracy of performing three types of tasks on two datasets. It can be seen that our method maintains the effect of the model better. In addition, the method achieves the effect of classical GNN, and even outperforms classical GNN on some tasks.

TABLE IV. ACCURACY OF INFERENCE RESULTS ON THREE TYPES OF TASKS (LP/NC/EC)

	Wikipedia			Reddit		
	LP	***NC***	***EC***	***LP***	***NC***	***EC***
GAT	87.34	-	-	92.14	-	-
TGAT	88.14	-	-	92.92	-	-
TGN	89.51	-	-	92.56	-	-
APAN	90.74	-	-	94.34	-	-
APAN-Re	87.42	99.81	99.81	97.76	99.91	99.91
distributed graph inference	87.03	99.54	99.28	97.01	98.39	98.05

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a distributed graph inference computing framework, aiming to use GNN models in the form of Encoder-Decoder for online deployment and real-time inference in distributed environments. The experimental results show that the upper limit of inference calculation has been greatly improved. It has also achieved good results in the timeliness of inference, and we believe that it can meet the actual needs in the case of limited resources. Furthermore, as graph iterative inference proceeds, our method can maintain the model's performance on three classes of tasks. In the future, we can extend the algorithm to adapt to more kinds of GNN models. And consider optimizing the communication overhead in a distributed environment, so that the system can have stronger real-time inference computing capabilities.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (2020AAA0103500).

REFERENCES

[1] Skardinga J, Gabrys B, Musial K. Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey[J]. IEEE Access, 2021.

[2] Zhang X C, Wu C K, Yang Z J, et al. MG-BERT: leveraging unsupervised atomic representation learning for molecular property prediction[J]. Briefings in Bioinformatics, 2021.

[3] Sahu S, Mhedhbi A, Salihoglu S, et al. The ubiquity of large graphs and surprising challenges of graph processing[J]. Proceedings of the VLDB Endowment, 2017, 11(4): 420-431.

[4] Sahu S, Mhedhbi A, Salihoglu S, et al. The ubiquity of large graphs and surprising challenges of graph processing: extended survey[J]. The VLDB Journal, 2020, 29(2): 595-618.

[5] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[6] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[7] Xu D, Ruan C, Korpeoglu E, et al. Inductive representation learning on temporal graphs[J]. arXiv preprint arXiv:2002.07962, 2020.

[8] Dwivedi V P, Joshi C K, Laurent T, et al. Benchmarking graph neural networks[J]. arXiv preprint arXiv:2003.00982, 2020.

[9] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[J]. HotCloud, 2010, 10(10-10): 95.

[10] Gonzalez J E, Xin R S, Dave A, et al. Graphx: Graph processing in a distributed dataflow framework[C]//11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14). 2014: 599-613.

[11] Cormen T H, Goodrich M T. A bridging model for parallel computation, communication, and I/O[J]. ACM Computing Surveys (CSUR), 1996, 28(4es): 208-es.

[12] Malewicz G, Austern M H, Bik A J C, et al. Pregel: a system for large-scale graph processing[C]//Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. 2010: 135-146.

[13] Gonzalez J E, Low Y, Gu H, et al. Powergraph: Distributed graph-parallel computation on natural graphs[C]//10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12). 2012: 17-30.

[14] Chen Q, Bai S, Li Z, et al. GraphHP: A hybrid platform for iterative graph processing[J]. arXiv preprint arXiv:1706.07221, 2017.

[15] Wang Z, Gu Y, Bao Y, et al. Hybrid pulling/pushing for i/o-efficient distributed and iterative graph computing[C]//Proceedings of the 2016 International Conference on Management of Data. 2016: 479-494.

[16] Tang J, Xu M, Fu S, et al. A scheduling optimization technique based on reuse in spark to defend against apt attack[J]. Tsinghua Science and Technology, 2018, 23(5): 550-560.

[17] Sankar A, Wu Y, Gou L, et al. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks[C]//Proceedings of the 13th International Conference on Web Search and Data Mining. 2020: 519-527.

[18] Rossi E, Chamberlain B, Frasca F, et al. Temporal graph networks for deep learning on dynamic graphs[J]. arXiv preprint arXiv:2006.10637, 2020.

[19] Kumar S, Zhang X, Leskovec J. Predicting dynamic embedding trajectory in temporal interaction networks[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 1269-1278.

[20] Wang X, Lyu D, Li M, et al. APAN: Asynchronous Propagation Attention Network for Real-time Temporal Graph Embedding[C]//Proceedings of the 2021 International Conference on Management of Data. 2021: 2628-2638.

PEM: A Parallel Ensemble Matching Framework for Content-based Publish/Subscribe Systems

Weidong Zhu[†], Yufeng Deng[§], Shiyou Qian [§*], Jian Cao[§] and Guangtao Xue[§]
[†]*Xuzhou University of Technology, Jiangsu, China.* [§]*Shanghai Jiao Tong University, Shanghai, China.*
[*]*Corresponding author, Email: qshiyou@sjtu.edu.cn*

***Abstract*—Content-based publish/subscribe systems are an effective paradigm for implementing on-demand event distribution. Each event needs to be matched against subscriptions to identify the target subscribers. To improve the matching performance, many novel data structures have been proposed. However, the predicates included in subscriptions are handled the same way in most existing data structures, which is not efficient given the matching probability of predicates. In this paper, we propose a parallel ensemble matching framework called PEM, which uses multiple algorithms with complementary behavior on predicate matching probabilities. To achieve the performance balance of parallel matching, we design an elastic subscription classification method. We implement a prototype of PEM based on two existing algorithms. The experiment results show that PEM improves the matching performance by 43%.**

I. Introduction

In the face of data explosion, fine-grained data distribution services are required in many fields. For example, the stock market generates massive data every day, for which investors need to subscribe and receive information on events of interest [1] [2]. Another example is an intelligent transportation scenario, where a large number of devices are deployed to collect a large volume of data. Likewise, drivers need a mechanism to obtain timely congestion and accident information specific to their driving route [3]. These applications motivate the need for an efficient way of propagating data from publishers (sources) to subscribers (destinations).

Content-based publish/subscribe (pub/sub) systems are an effective paradigm for implementing on-demand event distribution. To express their interest in data, subscribers first define subscriptions that usually contain multiple predicates [4] and then issue them to the broker (server). The publisher generates events consisting of multiple attribute-value pairs and sends them to the broker. For each event, the broker needs to match it with subscriptions to identify the target subscribers to whom the event information should be forwarded. In this way, publishers and subscribers are loosely coupled, which is the most attractive feature of content-based pub/sub systems [5].

Obviously, event matching is a key component in a content-based pub/sub system. Given a high-dimensional space, an event represents a point and a subscription represents a rectangle. Event matching is essentially a point enclosure search problem in nature, which is expensive in high-dimensional spaces. To make matters worse, when the number of subscriptions is large, the matching performance degrades, becoming a potential performance bottleneck.

To improve matching performance, many new data structures for storing subscriptions have been proposed, such as trees [6] [7] [8] [9], tables [10] [11] [12] and bloom filters [13] [14]. These novel data structures support efficient event matching. However, most existing structures index predicates in the same way, regardless of their matching probability. As verified in [15], most matching algorithms suffer from predicate matching probabilities. Increasing or decreasing the matching probability will result in performance degradation. One problem of the matching algorithm with fluctuating performance is that it cannot guarantee fast and stable data distribution services.

In this paper, we propose a parallel ensemble matching algorithm called PEM, which aims to improve and stabilize the matching performance using a multi-thread strategy. Similar to COMAT [16] which builds a library with multiple behavior-complementary matching algorithms, the basic idea of PEM is to leverage each algorithm by indexing each subscription in the appropriate algorithm. When matching events, all algorithms run in parallel, one thread per algorithm. When designing PEM, two points need to be addressed. First, we need to classify subscriptions based on the behavior of each algorithm on the predicate matching probability. Second, to prevent the overload of a specific algorithm, we need to establish a dynamic feedback mechanism to ensure the performance balance between multiple algorithms. Considering the characteristics of algorithms and the balance of threads, we design an elastic subscription classification method in PEM.

We implemented a prototype of PEM and conducted extensive experiments to evaluate its effectiveness and performance. The experiment results well verify the ability of PEM to improve and stabilize the matching performance. Compared with COMAT [16] and the other two baselines REIN [12] and TAMA [10], the matching time of PEM is improved by 43%, 55% and 47% respectively on average.

The main contributions of this paper are as follows:

- We propose an effective parallel ensemble matching framework called PEM to take advantage of multiple algorithms.
- We design an elastic subscription classification mechanism to maintain the performance balance between multiple threads.

DOI reference number: 10.18293/SEKE2022-051

- We implement a prototype of PEM and evaluate its effectiveness and performance through extensive experiments.

The remainder of this paper is organized as follows. We briefly discuss the related work in Section II. We describe the design details of PEM in Section III. Section IV elaborates the implementation of PEM. Section V presents and analyzes the experiment results. We discuss and conclude the paper in Section VI and VII respectively.

II. Related Work

In this section, we review the matching algorithms along two lines: sequential algorithms and parallel algorithms.

A. Sequential Matching Algorithms

Most existing matching algorithms were initially proposed as sequential, such as TAMA [10], MO-Tree [6], OpIndex [11], HEM [17] and REIN [12]. To achieve high matching performance, an efficient data structure for indexing subscriptions is necessary and critical for the matching algorithm. Classical data structures include matching trees [18] [19] [7], matching tables [20] [10] [11], binary decision diagrams [21] [22] and bloom filters [14] [13]. The underlying data structure of the matching algorithm is responsible for maintaining subscriptions (inserts, deletes and updates) and supporting event matching. In most existing matching algorithms, predicates are treated in the same way, regardless of their matching probabilities.

Liao et al. proposed a parallelization method called PhSIH to optimize the performance and stability of the sequential matching algorithm [23]. They parallelize three sequential algorithms using PhSIH, namely TAMA [10], OpIndex [11] and REIN [12]. To achieve a good parallelization effect, the sequential matching algorithm should have three characteristics. First, the workload of matching an event can be divided into sub-tasks from a data structure perspective. Second, the workload of the sub-tasks should be uniform, and the existence of a few dominant sub-tasks should be avoided. Third, the synchronization cost between sub-tasks should be small.

B. Parallel Matching Algorithms

Since most matching algorithms are executed sequentially, they cannot effectively utilize the parallel computing power of the hardware. Taking advantage of hardware development, such as multi-core CPUs, FPGAs and GPUs, some parallel matching algorithms have been proposed [24] [25] [26]. These algorithms typically parallelize event matching using a divide-and-conquer strategy, i.e. dividing the entire subscription set into subsets and building data structures on these subsets. While the divide-and-conquer approach is straightforward, it has limitations in flexibility and memory consumption.

The composite matching framework called COMAT is similar to our work [16]. In COMAT, each subscription is maintained in the data structures of multiple algorithms. When matching events, the matching time of all algorithms is estimated, and the optimal one is selected for event matching. COMAT can take advantage of different algorithms, but it has two limitations. First, each subscription is stored multiple times, which is not memory efficient. Second, in each algorithm, the predicate matching probability is not considered to optimize performance and stability.

Different from existing solutions, our work considers the effect of predicate matching probability on algorithm performance and stability. Considering the complementary behavior of different algorithms on predicate matching probability, subscriptions are maintained in the appropriate algorithm of PEM to take full advantage of the algorithm. Therefore, PEM can effectively improve and stabilize matching performance on the basis of existing algorithms.

III. Design of PEM

A. Overview

Obviously, our goal is to improve the matching performance for content-based pub/sub systems. Building on existing work, the design of the PEM framework is inspired by the idea of ensemble learning [27]. First, PEM uses a variety of algorithms to perform event matching. Subscriptions are classified according to their matching probability and assigned to the appropriate algorithm. Second, since subscription classification implies data parallelism, PEM allocates a thread to each algorithm to achieve parallel matching. The combination of subscription classification and parallel matching can greatly facilitate and stabilize matching performance.

1) Subscription Classification: As discussed in the work [15], most matching algorithms suffer from predicate matching probabilities. Subscriptions often contain multiple predicates with different matching probabilities. However, almost all existing matching algorithms treat predicates in their underlying data structures in the same way, regardless of the difference in matching probabilities. Intuitively, for subscriptions containing predicates with low matching-probability, it is efficient to use forward matching methods, such as TAMA [10] and OpIndex [11]. On the other hand, it is more efficient to use backward matching methods such as REIN [12] and GEM [28] to handle subscriptions with high matching probability. Therefore, we explore the idea of leveraging multiple algorithms with complementary behaviors in matching probability to improve performance.

2) Parallel Matching: Most existing matching algorithms are single-threaded. To speed up event matching, we can continuously optimize the performance of the single-threaded matching algorithm, but this performance improvement is generally difficult. With the development of computer hardware, multi-core CPUs and GPUs allow us to consider parallel matching, which has great potential to further improve matching performance. Therefore, we propose PEM based on subscription classification and multi-thread matching. PEM can greatly reduce the time to match events. In addition, PEM is beneficial to maintain the scalability of subscriptions and the stability of event matching under large-scale data.

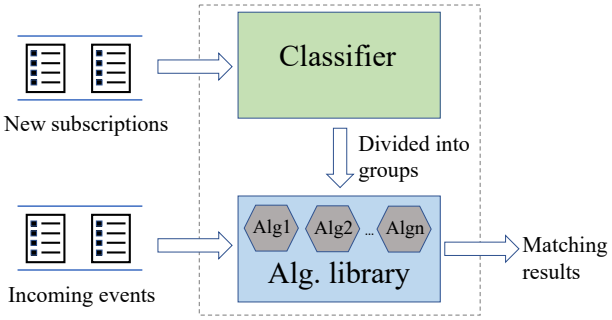


Fig. 1: The architecture of Ensemble Matching.

B. The Architecture of PEM

The architecture of PEM is shown in Fig.1, which consists of two modules: algorithm library and classifier. The library consists of multiple algorithms with complementary behaviors in terms of predicate matching probabilities. For each new subscription, the classifier estimates its matching probability and chooses the most appropriate algorithm to insert. Therefore, each subscription is maintained by an optimal matching algorithm. When a new event arrives, all algorithms in the library are executed in parallel and their matching results are aggregated.

1) Algorithm Library:

The basis for implementing PEM is to build a library of multiple matching algorithms. To do so, we give three criteria for choosing matching algorithms.

i) The algorithms in the library should complement each other. In other words, algorithms should have different performance behaviors in terms of predicate matching probability.

ii) Candidate algorithms should have similar overall matching performance, aiming to achieve good ensemble effects.

iii) For better generality, the algorithms in the library should support different subscription data models.

2) Classifier:

a) Quantification of Predicate Matching Probability: The matching probability of predicates in a subscription can be estimated by the average width of interval predicates and the number of predicates. An interval predicate has a low value and a high value that forms an interval. Other forms of predicates can be transformed to interval ones. For simplicity, we assume that events are uniformly distributed. Given a subscription containing K interval predicates, the average matching probability of the predicates can be estimated by

$$p = \frac{\sum_{i=1}^{K} w_i}{K} \tag{1}$$

where w_i is the width of the i^{th} interval predicate.

If the distribution of events is statistically available, the average matching probability of the predicates in the subscription can be calculated by

$$p = \frac{\sum_{i=1}^{K} \int_{l_i}^{h_i} p_e(x)\,dx}{K} \tag{2}$$

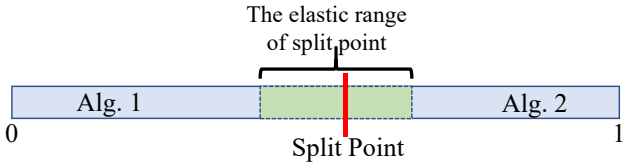


Fig. 2: Elastic classification of subscriptions considering algorithm characteristics and performance balance

where l_i and h_i represent the low value and high value of the i^{th} interval predicate respectively, and $p_e(x)$ is the probability density function of events.

b) Elastic Subscription Classification Method: When designing a subscription classification method for PEM, two points need to be considered for good parallelism. First, the correspondence between algorithm characteristics and subscription probabilities should be considered to assign subscriptions to appropriate algorithms. Second, since each algorithm runs using a single thread, skewed subscriptions can cause severe performance imbalances. Therefore, we propose an elastic subscription classification method.

For simplicity of discussion, we normalize the value domain of predicate matching probability in [0, 1]. Let L be the number of algorithms in the library. Since the algorithms in the library have complementary behaviors in terms of predicate matching probability, each algorithm has a range of applicable probability. The range of all algorithms collectively covers [0,1]. For example, assuming that the ith algorithm is optimal in the range [0.4, 0.5] and a subscription has a matching probability of 0.45, the subscription is assigned to the most suitable ith algorithm.

Given the L algorithms in the library, we need to compute $L-1$ split points SP_i $(1 \le i \le L-1)$ to separate the algorithms. SP_i represents the split point between algorithm i and algorithm $i+1$. The value of SP indicates a matching probability in [0, 1]. Considering algorithm characteristics and performance balance, the split point SP_i can fluctuate within a certain range $[R_{iMIN}, R_{iMAX}]$, which is called the elastic range. Specifically, SP_i can be calculated by:

$$SP_i = R_{i_{MAX}} - \frac{t_i \times (R_{i_{MAX}} - R_{i_{MIN}})}{T} \tag{3}$$

where t_i $(1 \le i \le L)$ is the matching time of algorithm i, $T = \sum_{i=1}^{L} t_i$ is the sum of the matching time of all algorithms, and $R_{i_{MAX}}$ and $R_{i_{MIN}}$ represent the elastic range of SP_i. Both $R_{i_{MAX}}$ and $R_{i_{MIN}}$ can be in the range of 0 to 1 and $R_{i_{MIN}} < R_{i_{MAX}}$. The initial value of SP_i is determined according to the characteristics of each algorithm in the library. The value of SP_i fluctuates within the elastic range $[SP_i \times (1-\alpha), SP_i \times (1+\alpha)]$. The value of α is set to 0.2 in the implementation.

We use the case of $L = 2$ to illustrate the concepts of split point and elastic range, as shown in Fig. 2. These two algorithms are suitable for subscriptions with different matching probabilities, where algorithm 1 (Alg. 1) is suitable for subscriptions with low matching probability, while algorithm 2 (Alg. 2) is suitable for subscriptions with high matching

Algorithm 1: Matching procedure of PEM

Require: an events e and the event window size ψ.

```
j++;
Match e using all algorithms in the library;
Aggregate the output of each algorithms to obtain the matching results;
if j = ψ then
    j = 0;
    Compute the average matching time t_i of each algorithms in the current window;
    Adjust the value of all split points SP_i in the next window according to Eq. (3);
end if
```

probability. The value of SP can fluctuate within an elastic range to maintain the performance of the two threads running the two algorithms separately.

The insertion process of PEM is straightforward. Given a new subscription, its matching probability p is first quantified according to Eq. (2). Then, the first split point $SP_i > p$ is found. The subscription is assigned to the ith algorithm and maintained in the corresponding data structure.

3) Matching Procedure of PEM: The matching procedure of PEM is shown in Algorithm 1. For each event, all algorithms in the library are used for matching, and their outputs are aggregated to obtain matching results. After matching ψ events in a time window, PEM adjusts the value of SP_i according to Eq. (3), aiming to maintain the performance balance among all algorithms based on the matching time of each algorithm in the library.

IV. Implementation

Similar to COMAT [16], we chose REIN [12] and TAMA [10] in the implementation to form the library. The two algorithms have similar overall performance and exhibit opposite behavior in terms of predicate matching probability. When matching events, TAMA uses a counter to record the predicate fulfillment condition for each subscription, while REIN uses a bitset to mark all unmatched subscriptions. Given an event, if a predicate evaluates to true, the counters of all subscriptions that contain the predicate are incremented by 1 in TAMA. Conversely, if a predicate evaluates to false, the bits representing all subscriptions that contain the predicate are marked in REIN. Therefore, REIN can achieve higher performance if more predicates evaluate to true, while TAMA may have lower performance, and vice versa.

Each algorithm in PEM is assigned to a thread. In the implementation, we need to balance the running time of REIN and TAMA. Since the split point of PEM is the criterion for subscription classification between REIN and TAMA, we design a dynamic feedback mechanism to maintain performance. After ψ events are matched in each window, the split point is adjusted according to the matching time ratio of REIN and TAMA. In the next window, the adjusted split point is applied. ψ is set to a small value of 20 in the implementation because the adjustment cost is almost negligible. Frequent tuning ensures that the two threads always reach a performance balance.

TABLE I: Parameters used in the experiments

Note	Description	Values
N	Number of subscriptions	**1M**, 2M, 3M, 4M,5M
M	Event size	**20**, 50, 100
K	Subscription size	5, **10**, 15, 20
W	Width of interval predicates	0.1, 0.3, **0.5**, 0.7, 0.9
SP	Split point of PEM	0.4, **0.5**, 0.6
ψ	Window size	20

V. Experiments

A. Experiment Setup

All the experiments were conducted on a server with 16 2.3GHz vCPUs and 32GB RAM, which runs Ubuntu 18.04 with Linux kernel 4.15.0-111. All code is written in the C++ language and compiled by g++ with version 7.5.0 and -O3 optimization.

We compare PEM with three baselines: COMAT [16], TAMA [10] and REIN [12]. According to the papers, the discretization level of TAMA is set to 17, and the number of buckets in REIN is set to 1000. The neural networks used in COMAT are implemented by the TensorFlow library with version 2.2.0-rc1. The matching time of COMAT is the sum of the matching time of the algorithm selected from REIN and TAMA and the prediction time of the neutral networks. The matching time of PEM is the time to execute REIN and TAMA in parallel using two threads. We run each experiment 10 times and obtain the average result.

In the experiments, by default, the number of subscriptions N is set to 1 million, the width of predicates W to 0.5, the number of predicates in subscriptions (subscription size) K to 10, and the number of attribute-value pairs in events (event size) M to 20. Subscriptions and events are randomly generated based on W, K and M. In each experiment, 500,000 events are matched and the average matching time is calculated. With large-scale datasets, we can better compare the performance of the tested matching algorithms. Table I lists the parameters used in the experiments. Bold values represent default settings.

B. Matching Performance

As shown in Fig. 3 and Fig. 4, PEM always performs best with different numbers of subscriptions N, followed by COMAT. TAMA appears to be more sensitive to N in terms of matching time and the standard deviation (Std) of matching time. From Fig. 3, we can see that TAMA outperforms REIN when the number of subscriptions is less than 3 million. REIN outperforms TAMA when the number of subscriptions becomes larger. When N = 1M, PEM achieves 43.7%, 68.6% and 46.9% performance improvements over COMAT, REIN and TAMA respectively on matching time. When N = 5M,

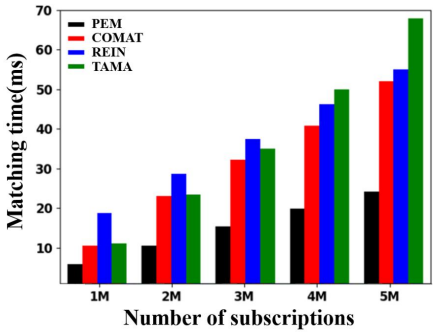


Fig. 3: Matching time with different numbers of subscriptions N

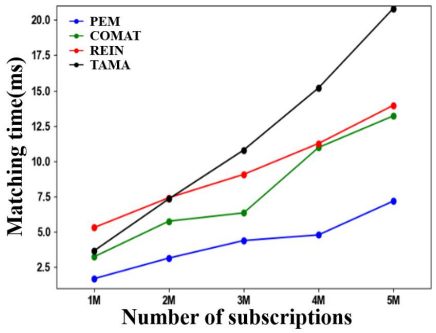


Fig. 4: Std of matching time with different numbers of subscriptions N

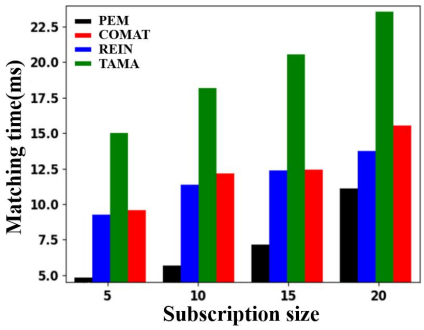


Fig. 5: Matching time with different subscription sizes K

the performance improvement of PEM over COMAT, REIN and TAMA is 53.3%, 55.8% and 64.2% respectively. Overall, PEM is around 40% faster than the second-best algorithm in all the experiments. In addition, PEM also has the lowest standard deviation of matching time, as shown in Fig. 4, which indicates that PEM has good performance stability.

In addition to N, the performance of the matching algorithm is also affected by several parameters, including K, M, and W. In the following subsections, we vary the settings of these parameters and evaluate their impact on the matching performance of PEM, COMAT, REIN and TAMA.

1) Effect of Subscription Size K: Figure 5 shows the effect of K on matching time. In the experiments, the subscription size is randomly generated in the range [1, K]. A larger K means that more predicates are stored in the structure of REIN and TAMA, and subscriptions have a lower matching probability. REIN is more susceptible to an increase in K as it needs to repeatedly mark more subscriptions as the matching probability of subscriptions decreases. When $K = 5$, PEM is 47.7%, 67.7% and 49.4% higher that COMAT, REIN and TAMA respectively. When $K = 20$, the improvement of PEM over COMAT, REIN and TAMA is 19.3%, 52.9% and 25.87% respectively. The performance of COMAT improves when K increases. This is because COMAT chooses TAMA more frequently than REIN as the number of predicates contained in each subscription increases. TAMA is better suited to subscriptions with a large number of predicates, reducing the performance difference between COMAT and PEM.

2) Effect of Event Size M: In this experiment, we set M from 20 to 100 and the results are shown in Fig. 6. A larger M means that there are more attribute-value pairs in events that the matching algorithm needs to process. We can find that REIN is more affected as M increases. This shows that under uniform distribution, REIN is more sensitive to event size. When $M = 20$, PEM achieves an improvement of 39.1%, 67.5% and 54.1% over COMAT, REIN and TAMA respectively. When $M = 100$, the improvement of PEM over COMAT, REIN and TAMA is 17.6%, 72.6% and 23.7% respectively.

3) Effect of the Width of Predicates W: The width of predicates is a key parameter that directly affects the performance of REIN and TAMA. REIN performs well at a wider width, while TAMA does the opposite. As shown in Fig. 7, when $W = 0.1$, PEM improves by -2.2%, 80.9% and 4.5% over COMAT, REIN and TAMA respectively. In this case, PEM, COMAT and TAMA perform almost identically, with COMAT slightly better than PEM. TAMA's performance is much better REIN. Conversely, when $W = 0.9$, PEM improves by 12.4%, 13.6% and 47.1% over COMAT, REIN and TAMA respectively. In this case, REIN significantly outperforms TAMA. When $W = 0.1$, almost all subscriptions are more suitable for TAMA. So, the two threads of PEM are unbalanced, and the results of PEM, COMAT and TAMA are very close. The situation is similar when $w = 0.9$. The results of PEM, COMAT and REIN are very close. When $W = 0.5$, PEM still outperforms the three baselines significantly. We believe that the performance of PEM is generally optimal when the predicate width varies.

C. Effectiveness of Feedback-based Adjustment Method

As shown in Fig. 8, the strategy of dynamically adjusting the split point consistently achieves the best performance when increasing the number of subscriptions. When $N = 200,000$, this strategy achieves 50.5%, 27.3%and 36.4% improvement over SP=0.4, SP=0.5 and SP=0.6 respectively. When $N = 1000,000$, the strategy achieves 31.2%, 6.6%and 20.8% improvement over SP=0.4, SP=0.5 and SP=0.6 respectively. We can see that the performance of SP=0.5 is better than SP=0.4 and SP=0.6. This is because the data is evenly distributed, with roughly an equal amount of data on both sides of 0.5. But the performance of dynamical adjustment is still better than SP=0.5, which reflects the effectiveness of the feedback-based update method in PEM.

VI. Discussion

The time it takes to insert a subscription in PEM is the sum of the subscription insertion time. Only one insertion is required per subscription in PEM. In our implementation, since the insertion time of REIN is much smaller than that of TAMA, the insertion time of PEM is close to that of TAMA. Subscription deletion is similar to subscription insertion.

Ideally, the dual-threaded parallel algorithm has a performance improvement of 100% compared to the single-threaded algorithm. In practice, we cannot have the exact same running

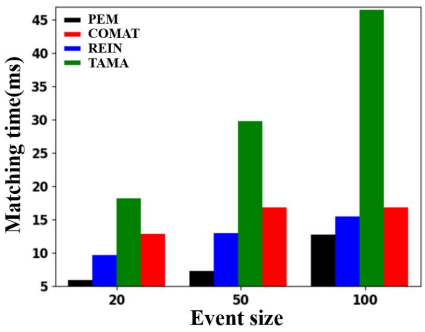


Fig. 6: Matching time with a different total number of attributes M.

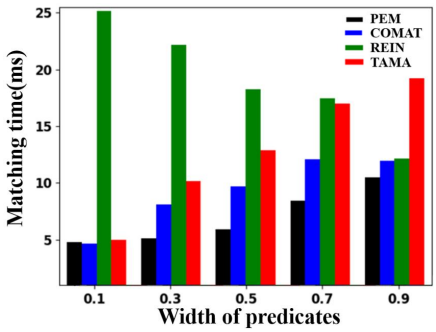


Fig. 7: Matching time with a different width of predicates W.

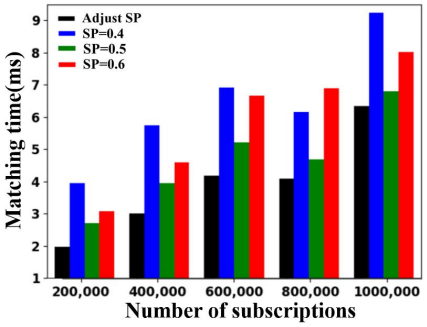


Fig. 8: Matching time with different split points.

time for both algorithms, and we also need time to classify the subscriptions. These factors take some time. In general, the performance improvement of PEM is still very significant.

VII. Conclusion

In this paper, we explore the idea of using multiple matching algorithms to improve and stabilize the performance of matching algorithms. The main idea of PEM is to classify subscriptions, choose the most appropriate algorithm for each subscription, and then perform parallel matching on multiple algorithms. To evaluate the performance of PEM, we conducted a series of experiments. The experiment results show that PEM can greatly improve and stabilize the matching performance under different parameter settings.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2019YFB1704400) and the National Natural Science Foundation of China (61772334).

References

[1] S. Qian, W. Mao, J. Cao, F. Le Mouël, and M. Li, "Adjusting matching algorithm to adapt to workload fluctuations in content-based publish/subscribe systems," in *IEEE INFOCOM*, 2019, pp. 1936–1944.

[2] T. Ding, S. Qian, J. Cao, G. Xue, and M. Li, "Scsl: Optimizing matching algorithms to improve real-time for content-based pub/sub systems," in *IEEE IPDPS*, 2020, pp. 148–157.

[3] N. Dasanayaka, C. Wang, D. Jayalath, and Y. Feng, "Publish-subscribe communications for V2I safety applications in intelligent transportation systems," in *IEEE VTC Fall*, 2019, pp. 1–6.

[4] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, 2001.

[5] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[6] T. Ding, S. Qian, J. Cao, G. Xue, Y. Zhu, J. Yu, and M. Li, "Mo-tree: An efficient forwarding engine for spatiotemporal-aware pub/sub systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 32, no. 4, pp. 855–866, 2021.

[7] S. Qian, J. Cao, Y. Zhu, M. Li, and J. Wang, "H-tree: An efficient index structurefor event matching in content-basedpublish/subscribe systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, no. 6, pp. 1622–1632, 2015.

[8] S. Ji and H. Jacobsen, "A-tree: A dynamic data structure for efficiently indexing arbitrary boolean expressions," in *ACM SIGMOD*, 2021, pp. 817–829.

[9] M. Sadoghi and H.-A. Jacobsen, "Be-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space," in *ACM SIGMOD*, 2011, pp. 637–648.

[10] Y. Zhao and J. Wu, "Towards approximate event processing in a large-scale content-based network," in *IEEE ICDCS*, 2011, pp. 790–799.

[11] D. Zhang, C.-Y. Chan, and K.-L. Tan, "An efficient publish/subscribe index for e-commerce databases," *VLDB Endowment*, vol. 7, no. 8, pp. 613–624, 2014.

[12] S. Qian, J. Cao, Y. Zhu, and M. Li, "Rein: A fast event matching approach for content-based publish/subscribe systems," in *IEEE INFOCOM*, 2014, pp. 2058–2066.

[13] S. Ji and H. Jacobsen, "Ps-tree-based efficient boolean expression matching for high dimensional and dense workloads," *VLDB Endowment*, vol. 12, no. 3, pp. 251–264, 2018.

[14] Z. Jerzak and C. Fetzer, "Bloom filter based routing for content-based publish/subscribe," in *ACM DEBS*, 2008, pp. 71–81.

[15] S. Qian, J. Cao, W. Mao, Y. Zhu, J. Yu, M. Li, and J. Wang, "A fast and anti-matchability matching algorithm for content-based publish/subscribe systems," *Computer Networks*, vol. 149, pp. 213–225, 2019.

[16] T. Ding, S. Qian, W. Zhu, J. Cao, G. Xue, Y. Zhu, and W. Li, "Comat: An effective composite matching framework for content-based pub/sub systems," in *IEEE ISPA*, 2020, pp. 236–243.

[17] W. Shi and S. Qian, "HEM: A hardware-aware event matching algorithm for content-based pub/sub systems," in *DASFAA*, 2022, pp. 277–292.

[18] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *ACM PODC*, 1999, pp. 53–61.

[19] M. Sadoghi and H.-A. Jacobsen, "Analysis and optimization for boolean expression indexing," *ACM Transactions on Database Systems (TODS)*, vol. 38, no. 2, pp. 1–47, 2013.

[20] A. Carzaniga and A. L. Wolf, "Forwarding in a content-based network," in *ACM SIGCOMM*, 2003, pp. 163–174.

[21] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith, "Efficient filtering in publish-subscribe systems using binary decision diagrams," in *IEEE ICSE*, 2001, pp. 443–452.

[22] G. Li, S. Hou, and H.-A. Jacobsen, "A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams," in *IEEE ICDCS*, 2005, pp. 447–457.

[23] Z. Liao, S. Qian, J. Cao, Y. Cao, G. Xue, J. Yu, Y. Zhu, and M. Li, "Phsih: A lightweight parallelization of event matching in content-based pub/sub systems," in *ICPP*, 2019, pp. 21:1–21:10.

[24] A. Farroukh, E. Ferzli, N. Tajuddin, and H.-A. Jacobsen, "Parallel event processing for content-based publish/subscribe systems," in *ACM DEBS*, 2009, pp. 1–4.

[25] K. Tsakalozos, M. Tsangaris, and A. Delis, "Using the graphics processor unit to realize data streaming operations," in *ACM Middleware Doctoral Symposium*, 2009, pp. 1–6.

[26] A. Margara and G. Cugola, "High-performance publish-subscribe matching using parallel hardware," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 1, pp. 126–135, 2014.

[27] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, pp. 5839–5847, 2018.

[28] W. Fan, Y. Liu, and B. Tang, "Gem: An analytic geometrical approach to fast event matching for multi-dimensional content-based publish/subscribe services," in *IEEE INFOCOM*, 2016, pp. 1–9.

MFGAN: A Novel CycleGAN-Based Network for Masked Face Generation

Weiming Xiong, Mingyang Zhong, Cong Guo, Huamin Wang, Libo Zhang*
College of Artificial Intelligence
Southwest University
Chongqing, China
lbzhang@swu.edu.cn

***Abstract*—In the post-epidemic era, Masked Face Recognition (MFR) is of great significance to our daily life, but it confronts a severe challenge of lacking real-world large-scale masked face datasets with identity labels. Moreover, mask enhances the diversity of face images and further improves the requirements for datasets. To address the above problem, we propose a novel CycleGAN-based masked face generation method MaskedFace-GAN (MFGAN), which is able to generate correct, authentic-looking and type-diverse masked face while ensuring the invariance of facial features. We design a three-stage training pipeline for MFGAN, which corresponds to three modules, respectively. Specifically, a facial feature detector is adopted to guide the model to generate the correct mask in the correct position. Then, by utilizing a mask binary segmentation module, the authenticity of generated images can be guaranteed. Lastly, with mask style encoder, the model can be optimized towards generating type-diverse masked faces. Finally, comparing with advanced masked face synthesis and generation methods comprehensively, our MFGAN achieves the best results. Then we apply the generated masked face datasets to MFR model training, which further proves the feasibility of training MFR models on generated datasets and the effectiveness and advancement of MFGAN compared with other state-of-the-art methods.**

***Index Terms*—Masked Face Generation, Image-to-Image Translation, Mask Style Encoder**

I. Introduction

In recent years, benefiting from the advancement of Convolutional Neural Networks (CNNs), face recognition has developed rapidly [1], [2]. Nowadays, people wear mask in reaction to global pandemics such as COVID-19, but it poses a great challenge to face recognition [3]. National Institute of Standards and Technology (NIST) found that, as most of the facial region are occluded by mask, the discriminative features that can be extracted by face recognition models are reduced, which leads to the degradation of the recognition performance of masked face [4]. However, taking off the mask for face recognition will increase the risk of infection, especially in crowded places such as airport [5]. Therefore, masked face recognition (MFR) is an urgent topic to research [3]. Furthermore, two possible solutions, occlusion robust face recognition (OFR) [6] and partial face recognition (PFR) [7] are not applicable, for they address different problems, as shown in Fig. 1. Mask occlusion is a kind of fixed position, large area, continuous and diverse occlusion, by contrast, random in OFR [6]. Moreover, masked face preserves the facial contour well, which cannot be guaranteed in PFR [7].

Fig. 1. Samples of occluded faces, partial faces and masked faces.

Despite its importance, MFR is still a challenging task due to the absence of large-scale real-world masked face datasets. In recent years, Generative Adversarial Networks (GANs) [8] has greatly promoted the development of image generation methods, and the use of them to generate dataset has gradually been widely adopted [9]. Therefore, GANs methods are naturally adopted to generate masked face, but they cannot be applied smoothly. Thereinto, masked face synthesis methods [10] directly overlay mask on face, which is prone to produce unnatural masked face. And masked face generation (MFG) methods confront two inevitable problems. Firstly, it is difficult to simultaneously generate authentic-looking mask and preserve the invariance of facial features [9], [11], [12]. Secondly, the generated mask types are not abundant and the methods cannot be applied to all kinds of datasets [13]–[15]. Recently, CycleGAN-based methods IAMGAN [15] and SimGAN [12] are specially proposed to generate masked face. However, they still suffer from the problems of incorrect wearing, sharpness distortion, etc, as shown in Fig. 2. Note that MFG is different from the other face generation tasks for the reason that the mask cover half of the face, and have various types, and are greatly influenced by the face posture, illumination and angle.

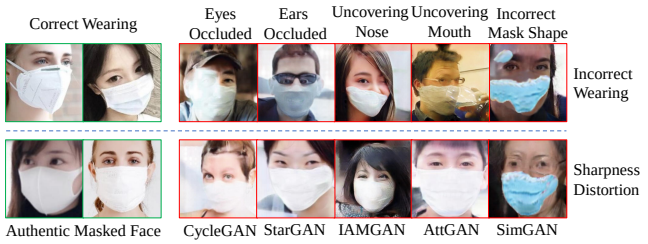


Fig. 2. Example results of some advanced masked face synthesis and generation methods reproduced in our experiments.

To address the aforementioned problems, we propose a novel CycleGAN-based masked face generation method

*Corresponding author: Libo Zhang (lbzhang@swu.edu.cn)
DOI reference number: 10.18293/SEKE2022-016

MaskedFaceGAN (MFGAN), which aims to generate correct, authentic-looking and type-diverse masked face images. In addition, in order to make the generated images meet the standard of training set, we consider more generation details, such as mask bandage, occlusion position, sharp edge, opaque, facial fidelity, fold feeling, multi-style, etc. Note that face recognition is a task involving major ethical issues, and this paper is devoted to the research of masked face generation. This paper only uses the official face datasets collected by legal means to conduct experiments, and only applies the generated datasets for MFR model training. Then we guarantee that the proposed method will not be applied to private face images, so as to protect personal privacy as much as possible. The generated samples on *FFHQ* [16] are shown in Fig. 3. Obviously, MFGAN can generate natural-looking mask on different ages, genders and complexion face.

Female Children Male

Fig. 3. Example results of generated masked face by proposed MFGAN.

In order to optimize MFGAN in three directions: correctness, authentic-looking and type-diversity, we design a three-stage training pipeline that introduces specific modules in different stages, detailed in Chapter III. (1) Firstly, in order to generate correct mask, we propose a facial feature detector, which can detect whether the facial features are occluded or not on the generated images according to the standard for correct mask wearing, so as to guide the model to generate the correct mask in the correct position. (2) Secondly, in order to generate authentic-looking mask, we propose a mask binary segmentation module to measure the fidelity of facial features in non-mask areas, so as to guide the model to generate mask without losing facial features as much as possible. (3) Thirdly, in order to generate type-diverse mask, we propose a mask style encoder, which can extract mask style code from the real-world referenced masked face, so as to guide the model to generate mask of corresponding style on the input face.

Finally, through qualitative and quantitative comparative experiments of the generated images, the proposed MFGAN achieves the best results. In addition, we further apply the generated images for MFR model training, the results show that the images generated by MFGAN is more suitable as training dataset for MFR model. The main contributions of this paper are listed as follows:

1) MFGAN is proposed to generate correct, authentic-looking and type-diverse masked face as a remedy of training dataset absence. We adopt it to generate two masked face datasets (*Masked-FFHQ*, *Masked-CelebA*) and publish them to facilitate future research [1].
2) We design a three-stage training pipeline. The facial detector ensures the correctness of masks, the mask binary segmentation module preserves the non-mask facial area, and the mask style encoder guides model to generate diverse styles of masks.

[1]https://github.com/MySky37/MySky.github.io

The outline of this paper is as follows. In Section II, we survey the related works. Then we introduce the proposed methods MFGAN in detail in Section III. Next, in Section IV, we conduct the experiments for generated images. Finally, we conclude the paper in Section V.

II. Related work

A. Masked Face Dataset

Wang et al. [17] proposed a Real-World Masked Face Dataset (*RMFRD*), including 5,000 images of 525 people with mask and 90,000 images of the same people without mask. Anwar et al. [10] proposed *MFR*2 dataset that includes 269 images of 53 politicians and celebrities from the Internet. However, the datasets proposed above are small-scale and not enough for MFR model training, but they are barely suitable as test sets. Therefore, we follow the previous work and adopt *RMFRD* as our test benchmark.

B. Masked Face Synthesis and Generation Method

Anwar et al. [10] proposed an open-source tool MaskTheFace, which is used to synthesize masked face. MaskedFaceNet [18] is a masked face synthesis method, which can synthesize masked face with different wearing postures, but its mask type is too single and not authentic-looking enough. Geng et al. [15] proposed a masked face generation method named Identity Aware Mask GAN (IAMGAN) with segmentation guided multi-level identity preserve module, which gained certain performance improvement compared to traditional CycleGAN [11]. However, the above methods have various disadvantages, as shown in Fig. 2. Following the previous work, we propose a novel CycleGAN-based masked face generation method MaskedFaceGAN (MFGAN).

III. Method

CycleGAN [11] is a classic unpaired image-to-image translation method, which is suitable for masked face generation, so we adopt it as the backbone of our MFGAN. Correctness, authenticity and type-diversity are three basic standards of masked face generation, so we specially design a three-stage training pipeline for MFGAN, as shown in Section III-A. In Section III-B, we propose a facial feature detector to guide the model to generate the correct mask in the correct position. In Section III-C, we propose a mask binary segmentation module to guide the model to generate authentic-looking mask without losing facial features as much as possible. In Section III-D, we propose a mask style encoder to guide the the model to generate diverse styles of masks.

A. Three-stage training pipeline

We design a three-stage training pipeline for MFGAN, as shown in Fig. 4, that is, we carry out gradual training for it, so that it can "learn" correctness first, then authenticity, and finally type-diversity. The computational complexity is also increasing gradually, detailed in Chapter IV.

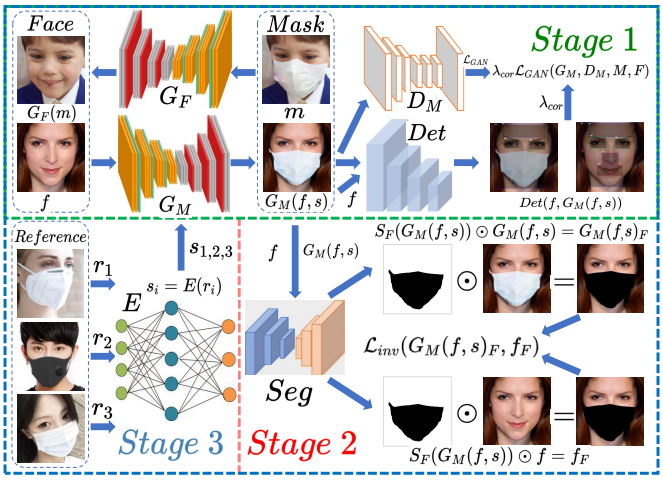


Fig. 4. A three-stage training pipeline of MFGAN. Each stage is represented by different colors and boxes. There are three image sets namely $Face$, $Mask$, and $Reference$. G_F and G_M are generators, D_M is a discriminator, Det is a facial feature detector, Seg is a mask binary segment module, and E is a mask style encoder. s_i is the mask style code of referenced masked face r_i. $G_M(f,s)_F$ and f_F represent the face with mask region removal.

B. Stage1: Generate Correct Mask

For masked face generation, the first requirement is to generate correct mask in the correct position. However, as shown in Fig. 2, some existing methods cannot guarantee the above premise well, and expose the problems of incorrect wearing, incorrect mask shape, etc. Therefore, we experimentalized and found that without additional supervision information, the generator is often prone to "make mistakes", as shown in Fig. 2. Therefore, we consulted the literature and learnt that the medical standard for wearing mask is to cover the mouth and nose, not eyes, and bandages should be hung on the ears. We also learnt that Yolov3 is a popular face detection method, and it can achieve good results in facial feature detection task. Therefore, based on a pretrained Yolov3 model, we construct a facial feature detector to check whether the mask in the generated image is worn correctly or not, as shown in Fig. 4 (Stage 1).

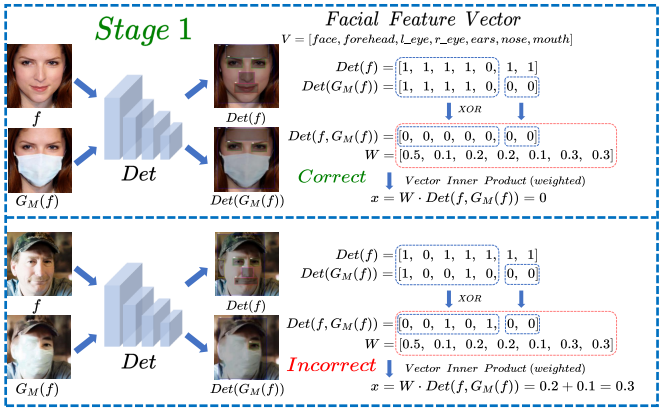


Fig. 5. Examples of detection result calculation. *XOR* represents exclusive OR operation, the same is 0 and the difference is 1. W means the penalty weight vector and x means the final penalty value.

The detection process is shown in Fig. 5 and described in detail as follows: Firstly, the proposed detector Det is used to detect facial features of face f and corresponding generated masked face $G_M(f)$, including face, eyes, nose, mouth, ears, forehead, etc. Secondly, the output item $D_i, i \in face, eye, mouth, ...$ with consistent feature detection results is 0, otherwise 1. For example, if eyes can be detected on face while not on masked face, which means the detection results are inconsistent and generated mask is incorrect, so the output is $D_{eye} = 1$. Then, all the outputs are formed into a vector $Det(f, G_M(f)) = [D_{face}, D_{eye}, D_{mouth}, ...]$, and further multiplied by the penalty weight vector W to obtain the final penalty value $x = W * Det(f, G_M(f))$. Thirdly, we input x into a revised adaptive correction function, which is an extended version of adaptive loss function [19], to get the adaptive correction factor λ_{cor}, the formula is as follows:

$$\lambda_{cor} = 1 + \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{x^2}{c^2|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right) \quad (1)$$

where $\lambda_{cor} = 1$ means correct mask wearing and $\lambda_{cor} > 1$ means incorrect. $\alpha \in R$ is a shape parameter that controls the specific form of the function. $c > 0$ is a scale parameter. Finally, we multiply the adaptive correction factor λ_{cor} with the adversarial loss of the discriminator $\mathcal{L}_{\text{GAN}}$ to obtain the corrected adversarial loss $\lambda_{cor}\mathcal{L}_{GAN}$. It provides additional supervisory information for discriminator to "learn" to distinguish true or false of the generated image according to the mask wearing condition. Then, based on the antagonistic game mechanism, by enhancing the distinctive ability of discriminator, the generator is forced to generate correct mask in correct position. The full objective of stage 1 is as follows:

$$\begin{aligned}\mathcal{L}_{GAN}(G_M, G_F, D_M, D_F) = &\lambda_{cor}\mathcal{L}_{GAN}(G_M, D_M, M, F) \\ &+ \mathcal{L}_{GAN}(G_F, D_F, F, M) \\ &+ \lambda\mathcal{L}_{cyc}(G_M, G_F)\end{aligned} \quad (2)$$

where $\mathcal{L}_{GAN}$ is the standard adversarial loss [8], $\mathcal{L}_{cyc}$ is the cycle consistency loss [11], and λ controls the relative importance of the two objectives.

C. Stage2: Generate Authentic-Looking Mask

Authentic-looking is the key element of masked face generation. Synthesis methods directly overlay the mask on face, which is prone to produce unnatural masked faces. Generation methods will inevitably lose facial features in the generation process, as shown in Fig. 6.

Fig. 6. Examples of facial features loss results of some generation models (CycleGAN, StarGAN, IAMGAN, AttGAN) from our experiments.

Geng et al. [15] adopted U-Net to guide masked face generation and achieved certain effect. Therefore, based on

a pre-trained U-Net, we construct a mask binary segmentation module to segment the mask area on the input face and the generated masked face, as shown in Fig. 4 (Stage 2). Given a masked face image $G_M(f)$, Seg predicts a binary segmentation map $S_F(G_M(f))$, where pixel value 0 and 1 represent the mask and non-mask region, respectively. Then we use the element-wise multiplication between $G_M(f)$ and $S_F(G_M(f))$, as well as f and $S_F(G_M(f))$, to obtain the image of mask area removal. Further, by calculating the similarity difference of the non-mask region before and after the generation, the local invariance loss function can be constructed as follows:

$$\mathcal{L}_{inv}(G_M(f)_F, f_F) = \mathbb{E}_{f\sim p_{\text{data}}(f)}\left[\|G_M(f)_F - f_F\|_2^2\right] \quad (3)$$

where $G_M(f)_F$ represents $G_M(f) \odot S_F(G_M(f))$ that means generated masked face of mask region removal, while f_F represents $f \odot S_F(G_M(f))$ that means input face of mask region removal. Then, we add $\mathcal{L}_{inv}$ into full objective of stage 2 and derive the following formula:

$$\begin{aligned}\mathcal{L}\left(G_M, G_F, D_M, D_F\right) = &\lambda_{cor}\mathcal{L}_{\text{GAN}}\left(G_M, D_M, M, F\right)\\ &+ \mathcal{L}_{\text{GAN}}\left(G_F, D_F, F, M\right)\\ &+ \lambda\mathcal{L}_{cyc}(G_M, G_F)\\ &+ \mu\mathcal{L}_{inv}(G_M(f)_F, f_F)\end{aligned} \quad (4)$$

where μ controls the relative importance of $\mathcal{L}_{inv}$.

D. Stage3: Generate Type-Diverse Mask

Type-diversity is an indispensable element in masked face datasets construction, because people wear masks in different types, colors and postures in the real world. However, most existing methods do not take it into consideration. Therefore, inspired by StarGANv2 [20] on diversified image translation between multiple domains, we propose a mask style encoder specially designed for masked face generation, which is used to instruct generator to generate mask in the direction of multi-style, as shown in Fig. 4 (Stage 3).

Generator: We extend the form of the input and output for generator G_M, which translates a face image f into a masked face image $G_M(f, s)$ according to domain-specific style code s provided by mask style encoder.

Mask Style Encoder: Given a referenced masked face r, our mask style encoder E extracts mask style code $s = E(r)$. E can produce diverse mask style codes using different referenced masked faces. This allows G_M to generate multi-style masked face reflecting the mask style s of the referenced masked face r. Our mask style encoder consists of a CNN with K output branches, where K is the number of mask style, and one of which is selected when training the corresponding mask domain. To make encoder more suitable for our task, we extend the pre-activation residual blocks to two ResStage blocks [2], and each block includes a Start ResBlock [2], a Middle ResBlock [2] and an End ResBlock [2], and output shape is changed correspondingly. Two ResStage block are also shared among all domains, followed by one specific fully connected layer for each domain. For the loss function, we adopt style reconstruction loss and style diversification loss, which are proposed by StarGANv2 [20].

Style reconstruction loss: It is designed to force the generator G_M to utilize the style code s when generating the masked face $G_M(f, s)$. The formula is as follows:

$$\mathcal{L}_{sty}(G_M, F, s) = \mathbb{E}_{f\sim p_{\text{data}}(f)}\left[\|s - E(G_M(f, s))\|_1\right] \quad (5)$$

Style diversification loss: It is designed to enable the generator G_M to produce diverse styles of masked face images according to the mask style codes. The formula is as follows:

$$\mathcal{L}_{ds}\left(G_M, F, s_1, s_2\right) = \mathbb{E}_{f\sim p_{\text{data}}(f)}\left[\|G_M\left(f, s_1\right) - G_M\left(f, s_2\right)\|_1\right] \quad (6)$$

where the target style codes s_1 and s_2 are produced by E, $s_i = E(r_i)$ for $i = 1, 2$. The goal of maximizing the loss is to force G_M to explore the image space and discover meaningful style features from the input masked face dataset for generating diverse masked face images. Then, we add $\mathcal{L}_{sty}$ and $\mathcal{L}_{ds}$ into full objective of stage 3 and derive the following formula:

$$\begin{aligned}\mathcal{L}\left(G_M, G_F, D_M, D_F\right) = &\lambda_{cor}\mathcal{L}_{\text{GAN}}\left(G_M, D_M, M, F\right)\\ &+ \mathcal{L}_{\text{GAN}}\left(G_F, D_F, F, M\right)\\ &+ \lambda\mathcal{L}_{cyc}(G_M, G_F)\\ &+ \mu\mathcal{L}_{inv}(G_M(f)_F, f_F)\\ &+ \lambda_{sty}\mathcal{L}_{sty}(G_M, F, s)\\ &- \lambda_{ds}\mathcal{L}_{ds}(G_M, F, s_1, s_2)\end{aligned} \quad (7)$$

where λ_{sty} and λ_{ds} are the weights of the corresponding items.

IV. Experiments

To evaluate the masked face generation performance of the proposed MFGAN and baselines, we made qualitative and quantitative comparative analysis and diversity display in Section IV-C. In Section IV-D, we compared the performance of MFR models trained on different generated datasets. Lastly, we conducted ablation study on the effectiveness of the proposed modules in MFGAN, detailed in Section IV-E.

A. Datasets and Implementation Details

Based on CycleGAN, MFGAN applies adaptive instance normalization (AdaIN) for up-sampling blocks in generator, and adds a convolution layer in discriminator. MFGAN is trained on *FFHQ* (as $Face$) and *RMFRD* (as $Mask$) with adam optimizer for 450K steps totally with batch size 1, and the training steps for three stages are 100K, 200K and 150K, respectively. Additionally, a facial feature detector Yolov3 is pretrained for 62.5K steps with adam optimizer on the detection version of *RMFRD*, and a mask binary segmentation module U-Net is pretrained for 20K steps with SGD optimizer on the segmentation version of *RMFRD*.

For the training of MFR model, we select a public large-scale face dataset *CelebA* and adopt three masked face generation and synthesis methods, MaskTheFace(MTF) [10], IAMGAN [15] and MFGAN, to construct three versions of *Masked-CelebA*[1], respectively.

B. Evaluation Metrics and Baselines

We evaluate the generated images quantitatively and qualitatively, and measure the masked face recognition (MFR) performance of the model trained on generated datasets. For quantitative evaluation, SSIM is used to measure the structural similarity between input face and generated masked face in non-mask area, PSNR is used for image quality evaluation, and FID is adopted to measure the data distribution distance between real-world masked face images and generated images. For qualitative evaluation, it mainly includes feature fidelity, mask transparency, mask type diversity, etc. For MFR performance evaluation, we choose *RMFRD* as test benchmark, and adopt verification accuracy, TAR@FAR=1e-3 and Rank-5 accuracy as evaluation metrics. For baselines, we compare MFGAN with two domain translation methods (CycleGAN [11] and StarGAN [9]), two facial attribute editing methods (AttGAN [13] and SaGAN [14]), and two CycleGAN-based methods (IAMGAN [15] and SimGAN [12]).

C. Comparison of Masked Face Image Generation Effect

In this section, we compare the masked face generated by our MFGAN and other generation and synthesis methods on *FFHQ*, quantitatively and qualitatively.

TABLE I
QUANTITATIVE COMPARISON RESULTS OF SOME ADVANCED MASKED FACE GENERATION METHODS.

Methods	SSIM	PSNR	FID
CycleGAN [11]	0.723	18.42dB	64.29
SimGAN [12]	0.701	16.42dB	83.19
SaGAN [14]	0.742	21.77dB	48.16
AttGAN [13]	0.781	24.81dB	35.77
StarGAN [9]	0.732	21.19dB	51.10
IAMGAN [15]	0.801	26.33dB	27.38
MFGAN	**0.838**	**29.52dB**	**21.73**

1) Quantitative Comparison: As shown in Table I, our MFGAN achieves the best results and outperforms the second best model IAMGAN by a large margin. The largest SSIM and PSNR indicate that, in masked face images generated by MFGAN, the feature information of the non-mask area is the best preserved, and the visual quality is better than others. Then combined with the smallest FID and the visual effect of the generated images, MFGAN can generate the most authentic-looking masked face images.

Fig. 7. Comparison with some state-of-the-art methods on masked face generation. MFGAN is able to generate authentic-looking masked face and preserve the facial features well.

2) Qualitative Comparison: For fair comparison, we randomly select four images from *FFHQ* and feed them into the compared models to generate the corresponding masked face images. From Fig. 7, we observe that the masked face generated by MFGAN are the most natural-looking, and have the best retention effect on the feature information of the non-mask area, while the baselines all have various shortcomings.

3) Diversity Display: Furthermore, most baselines cannot control the style of generated masked face, but our MFGAN can generate masked face with reference to real-world masked face, as shown in Fig. 8.

Fig. 8. Diversity display of masked face images generated by MFGAN.

Obviously, MFGAN can refer to many types of real-world masked faces and generate type-diverse masked faces. It not only proves the effectiveness of proposed mask style encoder, but also further proves the superiority of our MFGAN.

D. Comparison of MFR training effect on generated dataset

We adopt three methods to construct masked face datasets, and then apply them to the masked face recognition (MFR) training of four face recognition (FR) models, respectively.

TABLE II
TRAINING EFFECT OF FR MODELS ON *Mask-CelebA*, WHICH IS SYNTHESIZED OR GENERATED BY THE LEFTMOST METHODS.

Datasets	Methods	Acc	TAR@FAR=1e-3	Rank-5
MTF [10]	Softmax	77.2	65.1	61.2
	Triplet [21]	77.8	66.2	64.6
	CosFace [22]	78.4	67.9	66.5
	ArcFace [23]	78.5	67.9	66.3
IAMGAN [15]	Softmax	82.1	69.2	75.1
	Triplet	83.2	70.4	77.1*
	CosFace	83.6	71.5	72.9
	ArcFace	83.7*	71.6*	73.1
MFGAN	Softmax	89.2	75.9	80.3
	Triplet	90.1	76.7	**82.8(+5.7)**
	CosFace	90.4	78.1	78.1
	ArcFace	**90.9(+7.2)**	**78.4(+6.8)**	78.9

As shown in Table IV-D, we conclude the following results: (1) For verification task, face recognition (FR) models can extract distinguishing features from the full face to accurately judge whether the identities of two face are the same or not, but when confronting the masked face, the models can only extract a few features from non-mask areas, which easily leads to lower verification accuracy. (2) For recognition task, FR models can easily find the best matching identity from the face database, but when confronting the masked face, the models are more likely to be misled by similar faces to make wrong judgments, so the Rank-5 accuracy is relatively low. (3) However, with the same FR models, the training effect on the masked face dataset generated by MFGAN has achieved remarkable performance improvement, which fully proves the feasibility of training MFR models on the generated datasets and the effectiveness and advancement of our MFGAN.

E. Ablation Study

Finally, to analyze the function of different modules in MFGAN, we train three variants of it by removing $\mathcal{L}_{inv}$, λ_{cor}, and $\mathcal{L}_{sty}$, which controls the correctness, authenticity and type-diversity of the generated masked face. Additionally, we use CycleGAN as baseline, which lacks the above three modules simultaneously. The results are shown in Table III.

TABLE III
COMPARISON OF TRAINING EFFECT AND VISUAL QUALITY BETWEEN DIFFERENT VARIANTS OF MFGAN.

Methods	Performance			Visual Quality		
	Acc	TAR@FAR=1e-3	Rank-5	SSIM	PSNR	FID
CycleGAN	76.3	64.8	65.7	0.723	18.42dB	64.29
w/o $\mathcal{L}_{inv}$	80.3	68.5	70.8	0.747	20.44dB	58.46
w/o λ_{cor}	85.4	73.9	76.3	0.764	23.31dB	47.66
w/o $\mathcal{L}_{sty}$	87.2	75.3	80.1	0.813	27.31dB	25.98
All	**90.9**	**78.4**	**82.8**	**0.838**	**29.52dB**	**21.73**

Obviously, without $\mathcal{L}_{inv}$, MFGAN occurs serious performance degradation. Without λ_{cor}, MFGAN loses the ability to accurately control the generated position of mask. Without $\mathcal{L}_{sty}$, MFGAN cannot optimize in the direction of generating type-diverse masks. In general, lacking any modules will directly affect the model performance, which validates the effectiveness of the proposed modules and MFGAN.

V. CONCLUSION

In this paper, to alleviate the challenge of lacking large-scale real-world masked face datasets, we propose a novel CycleGAN-based masked face generation method Masked-FaceGAN (MFGAN), which enables to generate correct, authentic-looking and type-diverse masked face images. A three-stage training pipeline combined with facial feature detector, mask binary segmentation module and mask style encoder is designed to gradually optimize MFGAN. In addition, the masked face version of *FFHQ* and *CelebA* generated by MFGAN are publicly available to facilitate future research. Extensive experiments from quantitative, qualitative and diversity aspects have proved the practical significance and performance advantages of MFGAN and its corresponding modules. However, due to the lack of large area facial features, the masked face recognition (MFR) task is inherently difficult, and the performance of existing methods is still unsatisfactory. But the performance improvement of MFR models training on the datasets generated by MFGAN fully proves the feasibility of training MFR models on generated datasets. In the future, we will further enhance the robustness of MFGAN, and conduct in-depth research on MFR model.

ACKNOWLEDGMENT

This work is supported by the Fundamental Research Funds for the Central Universities (Nos. SWU021001 and SWU021002), the National Nature Science Foundation of China (No. 62106205), the Project of Science and Technology Research Program of Chongqing Education Commission of China (No. KJZD-K202100203 and KJQN202100207), and the Natural Science Foundation of Chongqing (Nos. cstc2021jcyj-msxmX0824 and cstc2021jcyj-msxmX0565).

REFERENCES

[1] Q. Zhang, Q. Cai, and J. Zheng, "Inspect defect of power equipment via deep learning method."
[2] I. C. Duta, L. Liu, F. Zhu, and L. Shao, "Improved residual networks for image and video recognition," in *2020 25th International Conference on Pattern Recognition*, 2021, pp. 9415–9422.
[3] A. Alzu'bi, F. Albalas, T. Al-Hadhrami, L. B. Younis, and A. Bashayreh, "Masked face recognition using deep learning: A review," *Electronics*, vol. 10, no. 21, p. 2666, 2021.
[4] M. L. Ngan, P. J. Grother, K. K. Hanaoka *et al.*, "Ongoing face recognition vendor test (frvt) part 6b: Face recognition accuracy with face masks using post-covid-19 algorithms," 2020.
[5] N. Damer, J. H. Grebe, C. Chen, F. Boutros, F. Kirchbuchner, and A. Kuijper, "The effect of wearing a mask on face recognition performance: an exploratory study," in *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 2020, pp. 1–6.
[6] D. Zeng, R. Veldhuis, and L. Spreeuwers, "A survey of face recognition techniques under occlusion," *arXiv preprint arXiv:2006.11366*, 2020.
[7] S. A. Abushanap, A. M. Abdalla, A. A. Tamimi, and S. Alzu'bi, "A survey of human face recognition for partial face view," in *2021 International Conference on Information Technology (ICIT)*. IEEE, 2021, pp. 571–576.
[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
[9] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.
[10] A. Anwar and A. Raychowdhury, "Masked face recognition for secure authentication," *arXiv preprint arXiv:2008.11104*, 2020.
[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
[12] S. Mumford, "Generation of realistic facemasked faces with gans," http://cs230.stanford.edu/projects_winter_2021/reports/70681837.pdf, 2021.
[13] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE transactions on image processing*, pp. 5464–5478, 2019.
[14] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*, 2019, pp. 7354–7363.
[15] M. Geng, P. Peng, Y. Huang, and Y. Tian, "Masked face recognition with generative data augmentation and domain constrained ranking," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 2246–2254.
[16] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.
[17] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei *et al.*, "Masked face recognition dataset and application," *arXiv preprint arXiv:2003.09093*, 2020.
[18] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19," *Smart Health*, 2021.
[19] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4331–4339.
[20] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020, pp. 8188–8197.
[21] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.
[22] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5265–5274.
[23] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.

Analysing Product Lines of Concurrent Systems with Coloured Petri Nets

Elena Gómez-Martínez Esther Guerra Juan de Lara

Universidad Autónoma de Madrid, Madrid, Spain
{MariaElena.Gomez, Esther.Guerra, Juan.DeLara}@uam.es

Abstract

Petri nets are a popular formalism to model and analyse concurrent systems. They can be combined with software product lines to support the specification of concurrent system families, like variants of controllers, process models, or configurations of flexible assembly lines. Specifically, a Petri net product line *(PNPL) comprises a (black and white) Petri net decorated with variability inscriptions, and a feature model controlling the derivation of admissible nets of the family. However, the derivable nets cannot be reconfigured at runtime, and the techniques to analyse properties of such reconfigurations are limited.*

To tackle these issues, we present a method to embed a PNPL into a standard Coloured Petri net. This embedding permits using the extensive simulation and analysis capabilities of powerful tools like CPN Tools, and enables the reconfiguration of the product nets at run-time. In this paper, we report on the translation of PNPLs into Coloured Petri nets, characterize the properties that can be analysed with this translation, and describe tool support on the basis of a case study in the area of flexible production systems.

1 Introduction

Petri nets [1] are a popular formalism to model and analyse concurrent systems. Their graphical nature makes modelling intuitive, and their strong theoretical basis enables powerful analysis possibilities. However, they are limited when the analysis of (possibly large) families of similar systems – like variants of process models [2], robots [3], configurations in flexible assembly lines [4] or reconfigurable manufacturing systems [5] – is required.

To solve this issue, in previous work, we proposed the notion of *Petri net product line* (PNPL) [6] as a compact way to specify families of net variants based on product line techniques. In essence, a PNPL combines a (black and white) Petri net annotated with presence conditions, and a feature model to express the allowed variability. Several Petri net analysis techniques have been lifted to enable the analysis of all nets of the family at once, instead of a case-by-case analysis [6]. However, the analysis is limited to structural properties (like marked graph, state machine and free-choice) and the reconfiguration between variants is not possible at run-time. This hinders the use of PNPLs in applications requiring dynamic reconfigurations, like self-adaptive cyber-physical systems [7, 8].

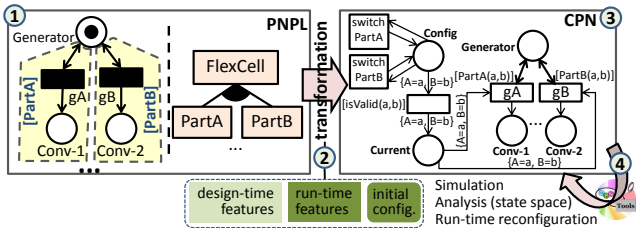


Figure 1. Overview of the approach.

To alleviate these issues, we propose to transform PNPLs into equivalent standard Coloured Petri nets (CPNs), as Fig. 1 shows. CPNs [9] extend Petri nets with data types, so that tokens can carry data, and arcs can query and produce tokens according to specified conditions. Our mapping synthesizes a CPN that explicitly represents the current configuration as a coloured token, emulates the presence conditions on the net elements via suitable arcs, and permits reconfiguring the system to a new feature configuration (cf. label 3 in Fig. 1). Prior to synthesizing the CPN (label 2), the user selects the features that may change at run-time, those fixed at design-time, and an initial configuration. Our mapping into CPNs enables the simulation of the running system and opens the door to useful analysis possibilities, e.g., based on model checking (label 4).

We have implemented the described mapping on an Eclipse plugin called TITAN. This tool supports the graphical modelling of PNPLs and the lifted analysis of structural properties. For this work, we have extended TITAN to transform a given PNPL into a CPN that can be simulated and analysed within CPN Tools [10].

2 Background

In this section, we introduce Petri nets (Sec. 2.1), PNPLs (Sec. 2.2) and CPNs (Sec. 2.3) using an example in flexible manufacturing systems.

2.1. Petri Nets

Petri nets [1] are a graphical formal notation to represent concurrent systems. A Petri net is a bipartite graph with two types of nodes: places (graphically depicted as circles) and transitions (drawn as rectangles). Places can be connected to transitions, and vice versa, via arcs. Petri nets have a *marking*, representing the distributed state of the net. The

DOI reference number: 10.18293/SEKE2022-015

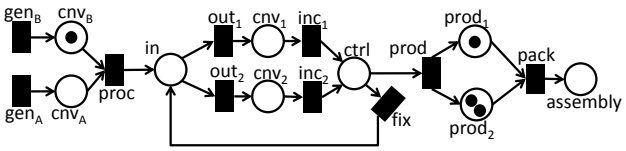


Figure 2. Petri net modelling an assembly line.

marking is given by sets of tokens (depicted as black circles within places) associated to each place.

Fig. 2 shows an example Petri net inspired by [6]. It represents an assembly line. Transitions gen_A and gen_B model generators of parts of types A and B, which transition proc processes and sends to any of two parallel conveyor belts (represented by places cnv_1 and cnv_2). After a quality control, transition fix sends the defective parts back. In a final step, two machines (represented by transitions prod and pack) process the parts, until they are assembled.

Petri nets can be simulated by the so-called token game. A transition is *enabled* if each input place has at least one token. Enabled transitions may fire at any moment. Firing a transition removes one token from each input place of the transition, and adds a token to each of its output places. In Fig. 2, transitions gen_A, gen_B and pack are enabled. Firing pack would add a token to place assembly, and would remove one token from prod_1 and prod_2.

2.2. Petri Net Product Lines

Petri nets are powerful to represent concurrent systems, but they are less suitable to capture families of similar systems in a compact way. For example, each possible configuration of a flexible assembly line – with different types of input parts, fabrication layouts, and output products – should be represented as a separate net. This is problematic if there are many configurations, the features of the assembly line can change, or properties of the whole family need to be analysed (e.g., can the assembly line manufacture a certain type of part in all configurations?).

PNPLs [6] combine Petri nets with product lines [11, 12] to tackle this problem. A PNPL comprises a feature model (FM) [13] describing the variability space, and a Petri net (called *150% net*) whose elements define presence conditions (PCs). The latter are boolean formulae over features of the feature model. Specific Petri net products can be *derived* from the PNPL by selecting a configuration. This derivation process removes all elements from the 150% net whose PC evaluates to false after substituting the selected configuration features by true, and the rest by false.

Fig. 3 shows an example PNPL with the possible configurations of a flexible assembly line. The feature model in Fig. 3(a) allows selecting one or more kinds of input parts (PartA, PartB), a fabrication layout (optional QualityControl, optional Parallel conveyor), and one or more kinds of output

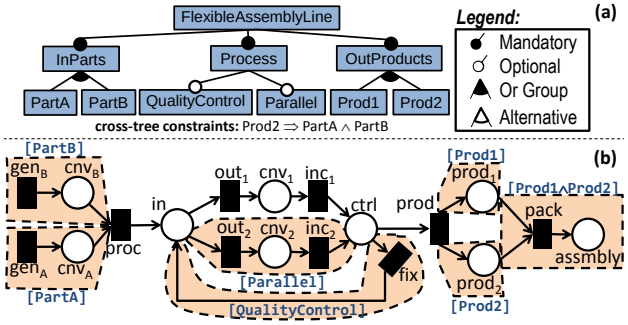


Figure 3. PNPL of a flexible assembly line. (a) Feature model. (b) 150% net.

products (Prod1, Prod2). A constraint forces that both types of input parts are selected to produce Prod2.

The 150% net in Fig. 3(b) has the same underlying net as Fig. 2, but its elements have PCs (shown in square brackets). We use dashed, coloured regions to assign the same PC to several elements. As an example, if a configuration does not select feature PartB, then transition gen_B, place cnv_B and their adjacent arcs would be removed from the derived product net.

2.3. Coloured Petri Nets

Coloured Petri nets (CPNs) [9] extend Petri nets by allowing tokens to carry data. The data structure is given by assigning a type (a *colour*) to the places. In CPN Tools, colours are specified with the Standard ML functional language [14], which supports defining datatypes like enumerations, product, union, list, and record types.

Arcs in CPNs are annotated with expressions, which can encapsulate complex computations. They may also include free variables, that need to be bound to suitable values found in the tokens. Transitions can have *guards*, which are boolean expressions that need to evaluate to true for the transition to be enabled. They can be used to test values from the variables bound in input arcs.

3 Analysing PNPLs with CPNs

Next, we present the transformation of PNPLs into CPNs (Sec. 3.1) and the analysis possibilities (Sec. 3.2).

3.1. Transforming PNPLs into CPNs

Our approach to analyse behavioural properties of PNPLs relies on CPNs. The rationale for this transformation is to be able to activate or deactivate the elements in the net structure (places, transitions, arcs) by means of expressions or guards in the CPN, according to the selected feature configuration. To achieve this, we use the feature model (FM) and the 150% net to guide the transformation.

3.1.1 Transforming the feature model.

To translate the feature model into a CPN, we initially generate two colour sets to encode a configuration. We create a generic type called FEATURE (a boolean); and a record colour set CONFIGURATION with n fields of type FEATURE, being n the number of features in the FM. Each FEATURE$_i$ stores whether the field is selected (true) or not (false) for a particular configuration. For the running example of Fig. 3, the created colour sets are the following:

```
colset FEATURE = BOOL;
colset CONFIGURATION = record
  INPARTS : FEATURE * PARTA : FEATURE * PARTB : FEATURE *
  PROCESS : FEATURE * QUALITYCONTROL : FEATURE *
  PARALLEL : FEATURE * OUTPRODUCTS : FEATURE * PROD1 : FEATURE *
  PROD2 : FEATURE;
```

Then, a place CONFIG with one token and the colour set CONFIGURATION is added to the CPN to represent any configuration, valid or not, generated from the FM. Values for tokens are extracted from an initial configuration set beforehand by the user. Moreover, to allow changing the feature selection at runtime, one transition Switch_Feature$_i$ per feature is included in the CPN. This transition is connected with the place CONFIG by two arcs: one input arc that reads the current value of the feature, and one outgoing arc switching its value. To reduce the size of the resulting CPN and consequently, its analysis time, we make the following optimization: if a feature needs to be selected in every valid configuration (i.e., it is mandatory) then we do not generate a transition to switch its value. In our running example, for instance, we do not generate such Switch transitions for features InParts, Process, OutProducts and FlexibleAssemblyLine. Similarly, the corresponding Switch transitions are not generated for the non-dynamic features that are fixed at design time (cf. step 2 in Fig. 1).

For the created transitions, the switched value is collected in a variable of type FEATURE. We also include two variables, named c and d, with type CONFIGURATION to store the current and the previous configuration, respectively. Therefore, the defined variables for the running example of Fig. 3 are the following:

```
var InParts, PartA, PartB, Process, QualityControl,
    Parallel, OutProducts, Prod1, Prod2 : FEATURE;
var c, d : CONFIGURATION;
```

The next step is to validate whether the current selection of features corresponds to a valid configuration. With this purpose, we incorporate a transition called isValid that reads a token from the place CONFIG. This transition has a guard, implemented as a function in the CPN ML language, which encodes the feature model as a propositional formula following the rules proposed in [15]. Therefore, only valid configurations satisfy this guard and enable the transition. Firing the transition passes the token with the variable c containing the value of all features for this valid configuration to the place CURRENT, and removes the old one. Thus,

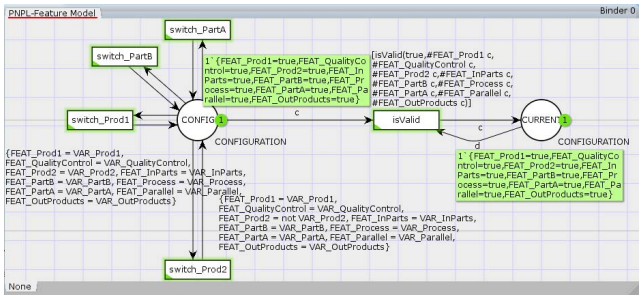


Figure 4. CPN for the feature model of the PNPL (some arc expressions omitted).

this place contains the token with the current configuration which will be used to activate and deactivate elements in the 150% net.

The next listing shows the function implementing the guard to validate configurations for the running example. It uses implication and bidirectional functions that we have defined explicitly, since they do not exist in ML.

```
fun implies (p,q) = not p orelse q;
fun iff (p, q) = ((not p orelse q) andalso (p orelse not q));
fun isValid (PARAM_FlexibleAssemblyLine, PARAM_Prod1,
     PARAM_QualityControl, PARAM_Prod2, PARAM_InParts,
     PARAM_PartB, PARAM_Process, PARAM_PartA, PARAM_Parallel,
     PARAM_OutProducts) =
  iff(PARAM_InParts,PARAM_FlexibleAssemblyLine) andalso
  iff(PARAM_Process,PARAM_FlexibleAssemblyLine) andalso
  iff(PARAM_OutProducts,PARAM_FlexibleAssemblyLine) andalso
  (PARAM_PartA orelse PARAM_PartB) andalso
  (implies(PARAM_QualityControl,PARAM_Process) andalso
    implies(PARAM_Parallel,PARAM_Process)) andalso
  (PARAM_Prod1 orelse PARAM_Prod2) andalso
  implies(PARAM_Prod2,(PARAM_PartA andalso PARAM_PartB));
```

Fig. 4 shows the fragment of the CPN resulting from the feature model depicted in Fig. 3(a), where features QualityControl and Parallel are static and hence there are no Switch transitions for them.

3.1.2 Transforming the 150% net.

The structural elements of the 150% net (places, transitions, arcs) have an almost direct translation into the CPN. The type of all the places is INT and they are marked according to the initial marking of the 150% net. To access the current configuration, every transition of the 150% net is connected by a bidirectional arc to the place CURRENT. The arc expression is the variable c, which stores the current configuration.

We model the PCs by means of arc expressions and transition guards. They allow to activate or deactivate elements of the net, since both determine if a transition is enabled (for a given marking). Specifically, the PCs of transitions are directly transformed into transition guards in the CPN. Regarding places, there is no way to activate or deactivate them in the CPN, hence, we emulate this behaviour by expressions in their input and output arcs. The PCs of arcs

are translated into *if-then-else* expressions, where the *if*-condition is the PC of the arc, the *then*-part is a new token to the output place, and the *else*-part is no token. For instance, the arc expression of the arc from transition pack to place assembly of the 150% net in Fig. 3 is:

```
if (#FEAT_Prod1 c andalso #FEAT_Prod2 c) then 1`1 else 0`1
```

where operator ♯ retrieves the field values within records. Overall, the expression adds a token only if the PC (Prod1∧Prod2) evaluates to true on the current configuration (given by c). As noted, PCs are parsed into the field names of the record colour set CONFIGURATION, and logical operators (and and or) are adapted to the ML language (andalso and orelse).

The resulting CPN model can be analysed using CPN Tools [10] in order to study behavioural properties. Fig. 5 shows the CPN obtained from the running example.

3.2. Analysing the PNPL

Most CPNs analyses are based on the *occurrence graph*: a graph-based representation of the state-space of reachable markings [9]. In PNPLs, this state-space represents all possible executions of the net, in all possible configurations allowed by the change of dynamic features. Once PNPLs are transformed into CPN, some of their properties that can be analysed are [10]:

Boundedness. A place is bounded if it can admit a limited number of tokens. CPN Tools reports the minimum and maximum bounds for each place. In PNPLs, the places associated to the feature model are bounded, and the bounds reported for the places in the 150% net are calculated for any possible configuration (no place in any configuration of our PNPL is bounded).

Home markings are those reachable from every reachable marking of the net. They may represent cyclic behaviours that might be desirable in all valid configurations (our PNPL lacks home markings).

Liveness is concerned with transitions staying active. CPN Tools reports dead markings (those where no firing is possible and the execution ends), live transitions (there is a firing sequence containing the transition in any reachable marking), and dead transitions (not enabled in any reachable marking). In PNPLs, the interest is on liveness of transitions of the 150% net, which refers to all possible configurations of dynamic features. Our PNPL lacks dead markings and dead transitions in any configuration.

Model checking allows formulating properties in temporal logics, to be checked on the state-space [16]. In PNPLs, these formulae can combine features of the feature model with properties of the markings in the 150% net. In our running example, we could check if, given a selection of static features, then for every configuration of the dynamic features, a token eventually reaches either prod_1 or prod_2 in every execution (so that products are produced). This is not the case in configurations with QualityControl.

4 Tool Support

We have built the tool TITAN (Tool for Petri net product line analysis) to support our approach. It is an Eclipse plug-in and uses the Eclipse Modeling Framework (EMF) [17] as the underlying modelling technology. It is available on `https://github.com/antoniogarmendia/titan`.

The tool integrates a graphical editor to define the 150% Petri net and its PCs. The editor is based on Sirius [18], a framework to create graphical modelling environments. TITAN also extends FeatureIDE [19] – a widely used plug-in in the field of product line engineering – to specify the feature model, the feature configurations and generate the product nets. TITAN supports the lifted analysis of structural properties of the PNPL, following the approach described in [6]. For this purpose, it relies on two libraries: the Sat4J solver [20] for solving boolean satisfaction problems (used to analyse properties state-machine, marked graph, free-choice and extended free-choice), and the JaCoP Java library as constraint programming (CP) solver [21] (used to analyse P- and T-invariants).

The architecture of TITAN is extensible via extension points with new analysis techniques and exporters to the input format of other Petri net tools. In addition to CPN Tools [10], which includes the presented transformation of PNPLs into CPNs, TITAN has exporters of the 150% net to GreatSPN [22], TimeNET [23] and WoPeD [24].

Fig. 6 shows TITAN with the running example. On the left, the Eclipse explorer contains the FeatureIDE project, which includes the PNPL of the running example, and the generated CPN file with its configuration file. The middle part shows the 150% net and its PCs. The right side displays the feature model. The result of the T-invariant analysis is presented on the bottom.

5 Related Work

Our proposal realises the notion of *dynamic software product line* (DSPL) for Petri nets. DSPLs [25] allow controlling the variability of adaptive systems at runtime. A DSPL can be seen as a system where the feature configurations correspond to system adaptations. Some authors analyse different aspects of DSPLs (but not for Petri nets). For instance, Sawyer et al. [26] use constraint solving to find the optimal configuration of self-adaptive systems, Olaechea et al. [27] employ trace checking to analyse the quality of service of all configurations of a DSPL, Göttmann et al. [28] translate DSPLs into timed automata to analyse the worst/best execution time of reconfiguration sequences, Ayala et al. [29] analyse DSPLs to predict the impact of recon-

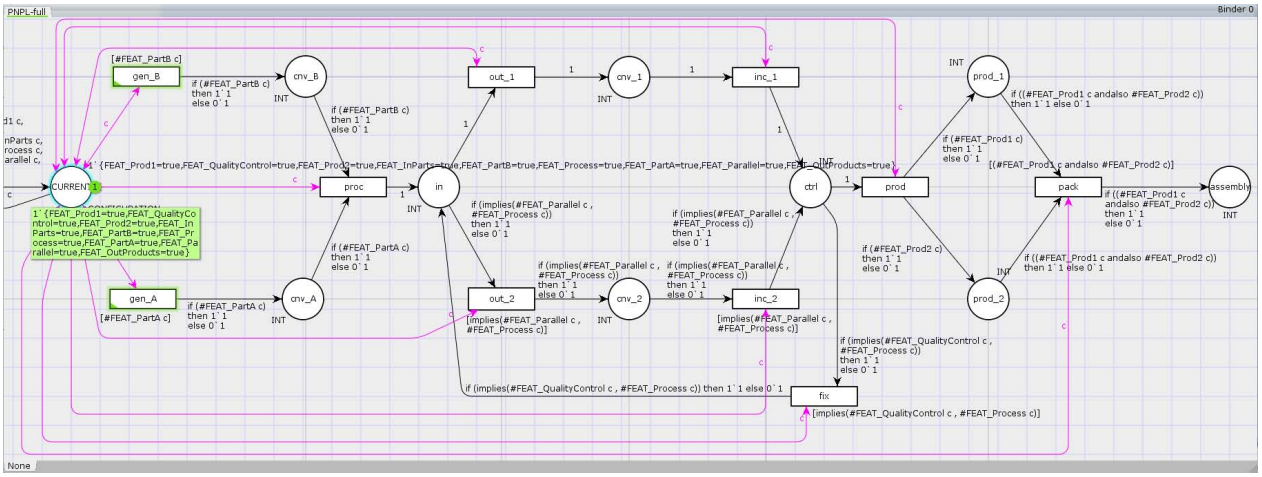


Figure 5. CPN for the 150% net and PCs of the PNPL.

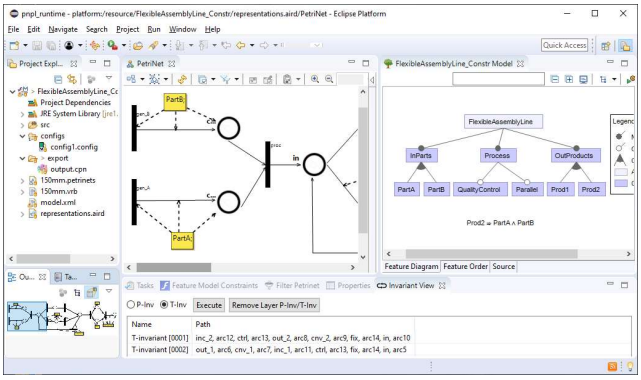


Figure 6. Screenshot of TITAN.

figurations on the system behaviour, and Quinton et al. [30] detect inconsistencies that may arise upon evolving a DSPL. Compared to them, our proposal relies on a translation of Petri nets into CPNs, which enables the analysis of properties based on model checking and the reachability graph.

Other works translate SPLs into Petri nets for analysis. For example, Martínez et al. [31] translate Orthogonal Variability Modeling (OVM) models capturing the variability of an SPL, into Petri nets which are analysed to uncover variants that do not appear in any SPL configuration. Their goal (analysing OVM models) is different from ours (analysing Petri net families), and the analysed properties also differ.

Closer to our approach, some works extend Petri nets with variability and variability-aware analysis techniques. *Feature nets* (FNs) [32] add variability to nets by attaching PCs to either transitions or arcs (but not to both at the same time as we do). The analysis of FNs is lifted to a variable reachability graph extended with PCs; instead we reuse proven standard analysis tools for CPNs out-of-the box. *Dynamic FNs* (DFNs) [32] extend FNs by enabling the firing of transitions to update the feature selection; instead, we use a feature model to decouple the net structure from its variability. *Adaptive Petri nets* [33] use modules to represent the variability of Petri nets. When selecting a configuration, the net is flattened by merging all the modules into a standard Petri net with inhibitor arcs. In our case, we capture the variability with a feature model, and the features are represented in the resulting CPN and can change at run-time.

Finally, Petri nets have been used to implement dynamic reconfigurations in different domains. Weyers [34] uses reference Petri nets (a kind of CPN) to model adaptive graphical user interfaces. Grobelna [35] represents reconfigurable modules of FPGAs as Petri nets, and model checks the satisfaction of the system requirements. Zhang [36] uses object-oriented CPNs with changeable structures to model reconfigurable manufacturing systems. We believe that our work could serve to define and provide analysis capabilities to these and other reconfigurable Petri net-based approaches.

6 Conclusions and Further Work

In this paper, we have presented a mapping from PNPLs into CPNs. The mapping enables the run-time adaptation of the net, and allows using the tooling and analysis methods available for CPNs. We have demonstrated the feasibility of the approach by an implementation atop the TITAN tool, which is able to target CPN Tools.

In the future, we plan to translate the analysis results from CPN Tools back into TITAN. We will also enrich PNPLs with time, to profit from the timing analysis capabilities of CPN Tools. Finally, we would also like to work on animating the token-game, lifting it to the PNPL level.

Acknowledgement

Work funded by the Spanish Ministry of Science (RTI2018-095255-B-I00) and the R&D programme of Madrid (P2018/TCS-4314).

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[2] M. L. Rosa, W. M. P. van der Aalst, M. Dumas, and F. Milani, "Business process variability modeling: A survey," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 2:1–2:45, 2017.

[3] S. García, D. Strüber, D. Brugali, A. D. Fava, P. Schillinger, P. Pelliccione, and T. Berger, "Variability modeling of service robots: Experiences and challenges," in *Proc. VaMoS*. ACM, 2019, pp. 8:1–8:6.

[4] Z. Nabi and T. Aized, "Modeling and analysis of carousel-based mixed-model flexible manufacturing system using colored Petri net," *Adv. in Mech. Eng.*, vol. 11, no. 12, pp. 1–14, 2019.

[5] J. Li, X. Dai, and Z. Meng, "Automatic reconfiguration of petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach," *IEEE Trans Autom. Sci. Eng.*, vol. 6, no. 1, pp. 156–167, 2009.

[6] E. Gómez-Martínez, J. de Lara, and E. Guerra, "Extensible structural analysis of Petri net product lines," *Trans. Petri Nets Other Model. Concurr.*, vol. XV, no. 12530, pp. 1–23, 2021.

[7] D. Perez-Palacin, J. Merseguer, and R. Mirandola, "Analysis of bursty workload-aware self-adaptive systems," in *Proc. ICPE*. ACM, 2012, pp. 75–84.

[8] R. Seiger, S. Huber, and T. Schlegel, "Toward an execution system for self-healing workflows in cyber-physical systems," *Softw. Syst. Model.*, vol. 17, no. 2, pp. 551–572, 2018.

[9] K. Jensen, *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, ser. EATCS Monographs on Theoretical Computer Science. Springer, 1992.

[10] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and CPN tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3–4, pp. 213–254, 2007, see also https://cpntools.org/.

[11] L. Northrop and P. Clements, *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[12] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering. Foundations, Principles and Techniques*. Springer-Verlag Berlin Heidelberg, 2005.

[13] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-90-TR-021, 1990.

[14] R. Milner, R. Harper, D. MacQueen, and M. Tofte, *The Definition of Standard ML*, revised edition ed. MIT press, 1997.

[15] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.

[16] E. M. Clarke, T. A. Henzinger, and H. Veith, "Introduction to model checking," in *Handbook of Model Checking*. Springer, 2018, pp. 1–26.

[17] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[18] Sirius, https://www.eclipse.org/sirius/.

[19] J. Meinicke, T. Thüm, R. Schröter, F. Benduhn, T. Leich, and G. Saake, *Mastering software variability with FeatureIDE*. Springer, 2017.

[20] D. L. Berre and A. Parrain, "The Sat4j library, release 2.2," *JSAT*, vol. 7, no. 2-3, pp. 59–6, 2010.

[21] K. Kuchcinski and R. Szymanek, "JaCoP - Java Constraint Programming solver," in *CP Solvers: Modeling, Applications, Integration, and Standardization*, 2013.

[22] E. G. Amparore, "Reengineering the editor of the greatspn framework," in *Proc. PNSE@Petri Nets*, ser. CEUR Workshop Proceedings, vol. 1372. CEUR-WS.org, 2015, pp. 153–170.

[23] A. Zimmermann, "Modelling and performance evaluation with timenet 4.4," in *QEST*, ser. LNCS, vol. 10503. Springer, 2017, pp. 300–303.

[24] T. Freytag and M. Sänger, "WoPeD - An Educational Tool for Workflow Nets," in *BPM (Demos)*, ser. CEUR Workshop Proceedings, vol. 1295. CEUR-WS.org, 2014, p. 31.

[25] S. O. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, "Dynamic software product lines," *Computer*, vol. 41, no. 4, pp. 93–95, 2008.

[26] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, "Using constraint programming to manage configurations in self-adaptive systems," *Computer*, vol. 45, no. 10, pp. 56–63, 2012.

[27] R. Olaechea, J. M. Atlee, A. Legay, and U. Fahrenberg, "Trace checking for dynamic software product lines," in *Proc. SEAMS@ICSE*. ACM, 2018, pp. 69–75.

[28] H. Göttmann, L. Luthmann, M. Lochau, and A. Schürr, "Real-time-aware reconfiguration decisions for dynamic software product lines," in *Proc. SPLC*. ACM, 2020, pp. 13:1–13:11.

[29] I. Ayala, A. V. Papadopoulos, M. Amor, and L. Fuentes, "ProDSPL: Proactive self-adaptation based on dynamic software product lines," *J. Syst. Softw.*, vol. 175, p. 110909, 2021.

[30] C. Quinton, M. Vierhauser, R. Rabiser, L. Baresi, P. Grünbacher, and C. Schuhmayer, "Evolution in dynamic software product lines," *J. Softw. Evol. Process.*, vol. 33, no. 2, 2021.

[31] C. Martínez, H. P. Leone, and S. M. Gonnet, "A petri net approach for representing orthogonal variability models," *International Journal of Computers & Technology*, vol. 9, no. 1, pp. 995–1003, 2013.

[32] R. Muschevici, J. Proença, and D. Clarke, "Feature nets: Behavioural modelling of software product lines," *Softw. Syst. Model.*, vol. 15, no. 4, pp. 1181–1206, 2016.

[33] C. Mai, R. Schöne, J. Mey, T. Kühn, and U. Assmann, "Adaptive Petri nets: A Petri net extension for reconfigurable structures," in *Proc. ADAPTIVE*. Springer, 2018, pp. 15–23.

[34] B. Weyers, "Formal description of adaptable interactive systems based on reconfigurable user interface models," in *The Handbook of Formal Methods in Human-Computer Interaction*. Springer International Publishing, 2017, pp. 273–294.

[35] I. Grobelna, "Model checking of reconfigurable FPGA modules specified by petri nets," *J. Syst. Archit.*, vol. 89, pp. 1–9, 2018.

[36] L. L. Zhang and A. Ittoo, "Development of an rms model based on colored object-oriented petri nets with changeable structures," in *Proc. IEEM*. IEEE, 2009, pp. 1719–1724.

Inter- and Intra-Series Embeddings Fusion Network for Epidemiological Forecasting

Feng Xie
National University of Defense Technology
Changsha, China
xiefeng@nudt.edu.cn

Zhong Zhang
National University of Defense Technology
Changsha, China
zhangzhong@nudt.edu.cn

Xuechen Zhao
National University of Defense Technology
Changsha, China
zhaoxuechen@nudt.edu.cn

Bin Zhou*
National University of Defense Technology
Changsha, China
binzhou@nudt.edu.cn

Yusong Tan
National University of Defense Technology
Changsha, China
ystan@nudt.edu.cn

ABSTRACT

The accurate forecasting of infectious epidemic diseases is the key to effective control of the epidemic situation in a region. Most existing methods ignore potential dynamic dependencies between regions or the importance of temporal dependencies and inter-dependencies between regions for prediction. In this paper, we propose an Inter- and Intra-**S**eries **E**mbeddings **F**usion **Net**work (SEFNet) to improve epidemic prediction performance. SEFNet consists of two parallel modules, named Inter-Series Embedding Module and Intra-Series Embedding Module. In Inter-Series Embedding Module, a multi-scale unified convolution component called Region-Aware Convolution is proposed, which cooperates with self-attention to capture dynamic dependencies between time series obtained from multiple regions. The Intra-Series Embedding Module uses Long Short-Term Memory to capture temporal relationships within each time series. Subsequently, we learn the influence degree of two embeddings and fuse them with the parametric-matrix fusion method. To further improve the robustness, SEFNet also integrates a traditional autoregressive component in parallel with nonlinear neural networks. Experiments on four real-world epidemic-related datasets show SEFNet is effective and outperforms state-of-the-art baselines.

KEYWORDS

deep learning; epidemiological forecasting; time series

1 INTRODUCTION

The outbreak of an epidemic will bring huge disasters to a region and even a country. The World Health Organization (WHO) estimates that influenza annually causes approximately 3-5 million severe cases and 290,000-650,000 deaths.[1] In recent years, the COVID-19 pandemic has spread to more than 200 countries and territories around the world,[2] and the number of infections and deaths in almost all affected countries is increasing at an alarming rate. Accurately forecasting epidemics plays an essential role in allocating healthcare resources and promoting administrative planning.

*Corresponding Author: binzhou@nudt.edu.cn
[1]https://www.who.int/en/news-room/fact-sheets/detail/influenza-(seasonal)
[2]https://covid19.who.int/
DOI reference number: 10.18293/SEKE2022-109

The epidemic prediction is similar to the multivariate time series forecasting task, but there are also significant differences. Multivariate time series forecasting methods inherently assume inter-dependencies among variables [21], while epidemic prediction needs to deal with unknown and complex patterns in the spread of epidemics and dynamic correlations between regions. The epidemic situation of a region at a certain time step is correlated with both its previous confirmed cases and other regions' epidemic situation. Therefore, two types of dependencies can be utilized in time series as demonstrated by Figure 1 and the epidemic time series modeling of regions can be decomposed into two parts:

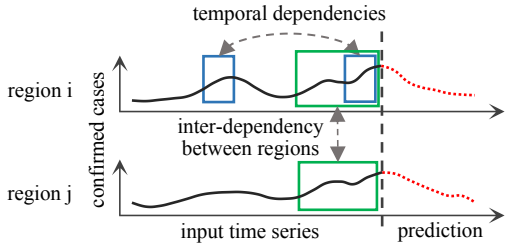


Figure 1: The blue boxes indicate the temporal dependency between time points, while the green boxes indicate the inter-dependency between regions.

- **Inter-series embedding modeling.** There are dynamic dependencies among regions/time series. When epidemics spread in different geographic regions, it is highly likely that similar progression patterns are shared among multiple regions owing to various factors [9] (e.g., similar geographic topology or climate), and these similar patterns can aid in prediction.
- **Intra-series embedding modeling.** There are temporal dependencies within a region/time series (e.g., seasonal influenza). The epidemic development trend of one region also can be distinguished from others, this is due to region-specific factors such as government intervention, healthcare quality, climate, etc.

To date, various methods have been proposed for epidemic forecasting, but they suffer from some limitations that are bad for performance. First, using vanilla Recurrent Neural Networks (RNNs) [2,

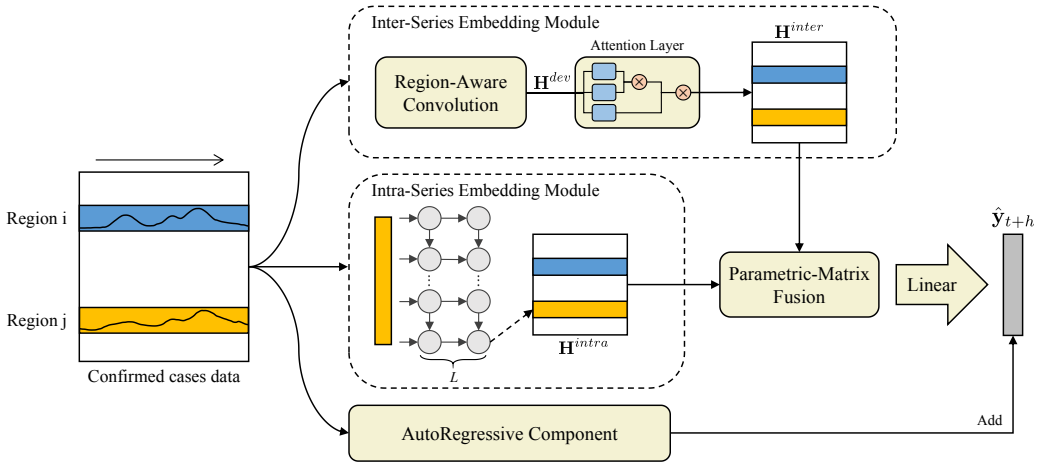


Figure 2: The overview of SEFNet. The original time series for each region are copied to three components: (1) Inter-Series Embedding module (top); (2) Intra-Series Embedding module (middle); and (3) AutoRegressive component (bottom).

10] or single-scale Convolution Neural Networks (CNNs) [9, 20] is hard to capture multi-scale and complex patterns, thus resulting in a certain degree of distortion, making it difficult to extract dynamic dependencies between time series. Second, some methods are dedicated to capturing dependencies between regions by introducing the "Attention Mechanism" [2, 9, 10], but these dependencies may misguide the final prediction because the progression patterns or data distribution of different regions is not fully consistent. Therefore, we believe both inter-series dependencies and intra-series dependencies jointly contribute to epidemic forecasting, and their influence degree on prediction results varies by region.

To tackle these challenges, we propose a novel deep learning model called Inter- and Intra-**S**eries **E**mbeddings **F**usion **Net**work (SEFNet) that extract inter- and intra-series embeddings through two parallel modules respectively and fuse them using parametric-matrix fusion [22]. To further improve the robustness, we also integrate autoregressive component parallel to the model. Our contributions are summarized below:

- We propose a new model that extracts inter-series correlations and intra-series temporal dependencies through two separate neural networks and uses parametric-matrix fusion to emphasize the importance of each information for epidemic prediction.
- We propose a multi-scale unified convolution component called Region-Aware Convolution that is capable of extracting local, periodic, and global patterns to better obtain feature representation and capture potential dependencies between regions.
- We conduct extensive experiments on four real-world epidemic-related datasets. The results show that our model achieves better performance than other state-of-the-art methods and demonstrates the effectiveness of each component.

2 RELATED WORK

There has been a large body of work focusing on epidemic forecasting in literature, including statistical models [8, 13, 18], compartment models [4, 19], and swarm intelligence models [3]. In recent years, deep learning models have shown excellent performance in various prediction tasks due to their powerful training and data-driven capabilities. CNNRNN-Res [20] is the first to apply deep learning for epidemiological prediction. Deng et al. [2] proposed Cola-GNN that treats regions as nodes in a graph and applies Graph Neural Networks (GNNs) to capture dependencies among regions. Jung et al. [10] proposed SAIFlu-Net that combines Long Short-Term Memory and self-attention to capture inter-dependencies between regions. Jin et al. [9] developed ACTs based on inter-series attention for COVID-19 forecasting. Cui et al. [1] designed a multi-range encoder-decoder framework for COVID-19 prediction. Nowadays, improving epidemic prediction is an open research problem to help the world mitigate the crisis that threatens public health.

3 THE PROPOSED METHOD

3.1 Problem Formulation

We formulate the epidemic prediction problem as a time series forecasting task. We have a total of N regions, and each region is associated with a time series input for a window T, where T is the length of historical observation data. Furthermore, we denote the epidemiology profiles $\mathbf{X} = [\mathbf{x}_{t-T+1}, ..., \mathbf{x}_t] \in \mathbb{R}^{N \times T}$ at time point t. An instance for region i is represented by $\mathbf{x}_{i:} = [x_{i,t-T+1}, ..., x_{i,t}] \in \mathbb{R}^T$. The goal of this task is to predict the epidemiology profile of the future time point $t + h$, where h is the horizon also called lead time. The proposed model SEFNet is shown in Figure 2. In the following sections, we introduce the building blocks of SEFNet in detail.

3.2 Intra-Series Embedding Module

The first module is Intra-Series Embedding module, which uses the historical information of time series to focus on the autocorrelation also called temporal dependencies of a single time series. In this work, we apply the Long Short-Term Memory (LSTM) [5] to capture temporal sequential dependency. The Recurrent Neural Networks (RNNs) are shown effective in sequence modeling and LSTM is a variant of RNNs, which can solve the vanishing gradients and exploding gradients problems in traditional RNNs [15]. Let D be

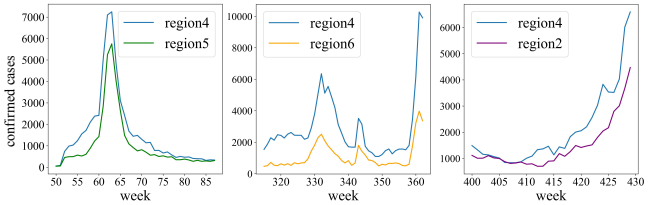
the dimension of the hidden state of LSTM, we use the original version of LSTM and formulate it as:

$$\mathbf{h}_{i,t} = \text{LSTM}(x_{i,t}, \mathbf{h}_{i,t-1}), \tag{1}$$

where $\mathbf{h}_{i,t}$ is the output representation of region i at time point t. For each region, we use the last output of LSTM as region's intra-series embedding $\mathbf{h}_i^{intra} \in \mathbb{R}^D$.

3.3 Inter-Series Embedding Module

The second module is Inter-Series Embedding module which focuses on dependencies between time series. First, we obtain temporal patterns through the proposed Region-Aware Convolution (RAConv), which is a multi-scale unified convolution component. Next, we feed the output of RAConv into an attention layer to generate embeddings of dynamic dependencies between regions.



(a) Local Pattern (b) Periodic Pattern (c) Global Pattern

Figure 3: The region 4 (blue line) has a dynamic pattern correlations with different regions within different time periods.

The correlation distribution is calculated based on the feature similarity between nodes [12]. The more accurate feature describes, the better performance of the attention layer can be improved. In epidemic prediction task, there are many similar progression patterns shared among regions, such as local patterns, periodic patterns, and global patterns. Figure 3 shows different temporal patterns of influenza case trends in different Health and Human Services (HHS) regions in the United States. Inspired by the Inception [16] in computer vision, we propose a multi-scale unified component called Region-Aware Convolution (RAConv) that can extract local, periodic, and global patterns simultaneously. The structure of RAConv is shown in Figure 4. RAConv consists of three branches that apply convolution blocks with different scales or different types, thus is capable of capturing multi-scale and more complex feature patterns. Each convolution block has K filters. The local pattern branch applies standard convolution with some small kernel sizes to extract local patterns in the time series through local mapping. The periodic pattern branch inspired by skip-RNN in LSTNet [11] applies dilated convolution that enables a large receptive field via dilation factor to capture the periodic pattern. Formally, the dilated convolution is a standard convolution applied to input with defined gaps. The global pattern branch applies standard convolution with the same size as T to extract time-invariant patterns of all time steps for regions [6] (e.g., time series uptrend in Figure 3c). We denote convolution filter in RAConv as $f_{1\times s,d}$ where s is kernel size and d is dilated factor. We empirically choose the kernel size s to {3,5,T}, and dilated factor d to {1,2}. We can get the local, periodic, and global features of region i by following equations:

$$\mathbf{h}_i^l = [\text{Pool}(\text{BN}(\mathbf{x}_{i:} \star \mathbf{f}_{1\times 3,1})); \text{Pool}(\text{BN}(\mathbf{x}_{i:} \star \mathbf{f}_{1\times 5,1}))], \tag{2}$$

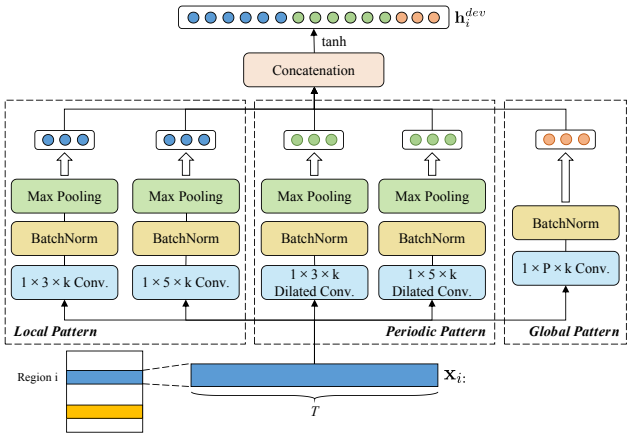


Figure 4: The structure of Region-Aware Convolution, which consists of three pattern branches.

$$\mathbf{h}_i^p = [\text{Pool}(\text{BN}(\mathbf{x}_{i:} \star \mathbf{f}_{1\times 3,2})); \text{Pool}(\text{BN}(\mathbf{x}_{i:} \star \mathbf{f}_{1\times 5,2}))], \tag{3}$$

$$\mathbf{h}_i^g = \text{BN}(\mathbf{x}_{i:} \star \mathbf{f}_{1\times T,1}), \tag{4}$$

where $\star$ is convolution operator, and $[;]$ is concatenation operation. $Pool(\cdot)$ is the Adaptive Max Pooling layer that can not only capture the most representative features, but also effectively reduce the amount of parameters. Adaptive Max Pooling is able to control the output size same as parameters P. $BN(\cdot)$ is the Batch Normalization [7] layer that normalize the data and speed up convergence. The convolution operation of $\mathbf{x}$ with $\mathbf{f}$ at step j is represented as:

$$\mathbf{x} \star \mathbf{f}_{1\times s,d}(j) = \sum_{i=1}^{s} \mathbf{f}_{1\times k}(i)\mathbf{x}(j - d \times i). \tag{5}$$

Next, we concatenate three patterns and apply an element-wise activation function (e.g., hyperbolic tangent):

$$\mathbf{h}_i^{dev} = \tanh([\mathbf{h}_i^l; \mathbf{h}_i^p; \mathbf{h}_i^g]). \tag{6}$$

For each time series, we execute the above process and get the intermediate matrix called $\mathbf{H}^{dev} \in \mathbb{R}^{N\times(P+1)K}$.

Due to the powerful feature extraction capability of the self-attention network, we apply a typical self-attention network inspired by the Transformer [17] to capture the dependencies among regions. Let A be the dimension of inter-series embedding. We can calculate attention distribution $\mathbf{A}$ and inter-series embedding matrix $\mathbf{H}^{inter} \in \mathbb{R}^{N\times A}$ by following equations:

$$\mathbf{A} = \text{softmax}((\mathbf{H}^{dev}\mathbf{W}^Q)(\mathbf{H}^{dev}\mathbf{W}^K)^T), \tag{7}$$

$$\mathbf{H}^{inter} = \mathbf{A}\mathbf{H}^{dev}\mathbf{W}^V, \tag{8}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V \in \mathbb{R}^{(P+1)K\times A}$ are the weight matrices that linearly map the $\mathbf{H}^{dev}$ to query, key, and value matrices.

3.4 Fusion

Directly concatenating or summing inter-series embedding and intra-series embedding will have the following problems: (1) **Inconsistent scale.** Since two feature embeddings come from different neural network modules, the structural differences of each module (e.g., activation function) will lead to inconsistent scales of feature embeddings; (2) **Different importance.** Two feature embeddings describe different feature information of time series so the

importance of two feature embeddings is very different in the process of epidemiology forecasting (e.g, temporal dependency is more significant for a region with periodic recurrence of an epidemic, although there may be similar development patterns to others). Therefore, to address these problems, we adopt parametric-matrix fusion [22] to adaptively control the flow of inter-series embedding and intra-series embedding and fuse them together:

$$\mathbf{H}^{fus} = [\mathbf{W}^{inter} \circ \mathbf{H}^{inter}; \mathbf{W}^{intra} \circ \mathbf{H}^{intra}], \tag{9}$$

where $\mathbf{H}^{fus} \in \mathbb{R}^{N \times (D+A)}$ is the ouput of fusion operation. $\circ$ is element-wise multiplication. $\mathbf{W}^{inter}$ and $\mathbf{W}^{intra}$ are the learnable parameters that adjust the degrees affected by inter-series embedding and intra-series embedding respectively.

3.5 Prediction

Due to the nonlinear characteristics of Convolutional, Recurrent and self-attention components, the scale of neural network output is not sensitive to input. Meanwhile, the historical infection cases of each region are not purely nonlinear, which cannot be fully handled well by neural networks. To address these drawbacks, we retain the advantages of traditional linear models and neural networks by combining a linear part to design a more accurate and robust prediction framework inspired by [11, 14]. Specifically, we adopt the classical AutoRegressive (AR) model as the linear component in a parallel manner. Denote the forecasting result of AR component as $\hat{\mathbf{y}}^{l}_{t+h} \in \mathbb{R}^{N}$ that can be calculated by following equation:

$$\hat{y}^{l}_{i,t+h} = \sum_{m=0}^{q-1} \mathbf{W}^{ar}_{m} x_{i,t-m} + b^{ar}, \tag{10}$$

where $\mathbf{W}^{ar}$ is the weight matrix and b^{ar} is the bias. q is the look-back window of AR that need be less than or equal to input window size T. Then, we feed the output after fusion operation to a dense layer to get the nonlinear part of the final prediction:

$$\hat{\mathbf{y}}^{n}_{t+h} = \mathbf{H}^{fus}\mathbf{W}^{n} + \mathbf{b}^{n}. \tag{11}$$

The final prediction of model is then obtained by summing the nonlinear part and the linear part got by AR component:

$$\hat{\mathbf{y}}_{t+h} = \hat{\mathbf{y}}^{n}_{t+h} + \hat{\mathbf{y}}^{l}_{t+h}. \tag{12}$$

In the training process, we adopt the Mean Square Error as the loss function that defined as:

$$\underset{\theta}{minimize} \, \|\mathbf{y}_{t+h} - \hat{\mathbf{y}}_{t+h}\|_2^2, \tag{13}$$

where $\mathbf{y}_{t+h} = [y_{1,t+h}, ..., y_{N,t+h}] \in \mathbb{R}^{N}$ is the true value at time point $t+h$, and θ are all learnable parameters in the model.

Table 1: Dataset statistics: min, max, mean, and standard deviation (SD) of patient counts; dataset size means the number of regions multiplied by the number of samples.

Datasets	Size	Min	Max	Mean	SD
Japan-Prefectures	47×348	0	26635	655	1711
US-Regions	10×785	0	16526	1009	1351
US-States	49×360	0	9716	223	428
Canada-Covid	13×717	0	127199	3082	8473

4 EXPERIMENTS

4.1 Datasets and Metrics

We prepare four real-world epidemic-related datasets as follows, and their data statistics are shown in Table 1.

- **Japan-Prefectures.** This dataset is collected from the Infectious Diseases Weekly Report (IDWR) in Japan,[3] which contains weekly influenza-like-illness statistics from 47 prefectures, ranging from August 2012 to March 2019.
- **US-Regions.** This dataset is the ILINet portion of the US-HHS dataset,[4] consisting of weekly influenza activity levels for 10 HHS regions of the U.S. mainland for the period of 2002 to 2017. Each HHS region represents some collection of associated states.
- **US-States.** This dataset is collected from the Center for Disease Control (CDC).[4] It contains the count of patient visits for ILI (positive cases) for each week and each state in the United States from 2010 to 2017. After removing a state with missing data we keep 49 states remaining in this dataset.
- **Canada-Covid.** This dataset is publicly available at JHU-CSSE.[5] We collect daily COVID-19 cases from January 25, 2020 to January 10, 2022 in Canada (including 10 provinces and 3 territories).

We adopt two metrics for evaluation that are widely used in epidemic forecasting, including the **Root Mean Squared Error (RMSE)** and the **Pearson's Correlation (PCC)**. RMSE measures the difference between predicted and true value after projecting the normalized values into the real range. PCC is a measure of the linear dependence between time series. For RMSE lower value is better, while for PCC higher value is better.

4.2 Methods for Comparison

We compared the proposed model with the following methods.

- **AR** The most classic statistical methods in time series analysis.
- **LRidge** The vector autoregression (VAR) with L2-regularization.
- **LSTNet [11]** A deep learning model that combines CNN and RNN to extract short- and long-term patterns.
- **TPA-LSTM [14]** An attention based LSTM network that employs CNN for pattern representations;
- **CNNRNN-Res [20]** A deep learning model that combines CNN, RNN, and residual links for epidemiological prediction.
- **SAIFlu-Net [10]** A self-attention based deep learning model for regional influenza prediction.
- **Cola-GNN [2]** A deep learning model that combines CNN, RNN and GNN for epidemiological forecasting.

4.3 Experimental Details

All programs are implemented using Python 3.8.5 and PyTorch 1.9.1 with CUDA 11.1 (1.9.1 cu111) in an Ubuntu server with an Nvidia Tesla K80 GPU. Our source codes are publicly available.[6]

Experimental setting. All datasets have been split into training set(50%), validation set(20%) and test set(30%). The batch size is set to 128. We use Min-Max normalization to convert data to [0,1] scale and after prediction, we denormalize the prediction value and use it

[3] https://tinyurl.com/y5dt7stm
[4] https://tinyurl.com/y39tog3h
[5] https://github.com/CSSEGISandData/COVID-19
[6] https://github.com/Xiefeng69/SEFNet

Table 2: RMSE and PCC performance of different methods on four datasets with horizon = 3, 5, 10. Bold face indicates the best result of each column and underlined the second-best. For RMSE lower value is better, while for PCC higher value is better.

Dataset			Japan-Prefectures			US-Regions			US-States			Canada-Covid		
			Horizon			Horizon			Horizon			Horizon		
Methods	Metrics		3	5	10	3	5	10	3	5	10	3	5	10
AR	RMSE		1705	2013	2107	757	997	1330	204	251	306	3488	4545	7154
	PCC		0.579	0.310	0.238	0.878	0.792	0.612	0.909	0.863	0.773	0.973	0.955	0.869
LRidge	RMSE		1711	2025	1942	870	1059	1270	276	295	324	3326	4372	7179
	PCC		0.308	0.429	0.238	0.878	0.792	0.612	0.909	0.863	0.773	0.975	0.957	0.868
LSTNet	RMSE		1459	1883	1811	801	998	1157	249	299	292	3270	6789	9561
	PCC		0.728	0.432	0.518	0.868	0.746	0.609	0.850	0.759	0.760	0.967	0.847	0.645
TPA-LSTM	RMSE		1142	1192	1677	761	950	1388	203	247	236	2731	3905	7671
	PCC		0.879	0.868	0.644	0.847	0.814	0.675	0.892	0.833	0.849	0.980	0.956	0.767
CNNRNN-Res	RMSE		1550	1942	1865	738	936	1233	239	267	260	6175	8644	9755
	PCC		0.673	0.380	0.438	0.862	0.782	0.552	0.860	0.822	0.820	0.659	0.589	0.475
SAIFlu-Net	RMSE		1356	1430	1527	661	871	1158	167	238	236	4409	7128	8514
	PCC		0.765	0.654	0.592	0.903	0.800	0.674	0.927	0.842	0.845	0.745	0.775	0.596
Cola-GNN	RMSE		1051	**1117**	1372	636	855	1134	167	202	241	2954	4036	7336
	PCC		0.901	0.890	0.813	**0.909**	0.835	0.717	0.933	0.897	0.822	0.986	0.975	0.882
SEFNet	RMSE	(↓)	**1020**	1123	**1319**	**618**	**821**	**1036**	**162**	**196**	**232**	**2157**	**3339**	**7079**
	PCC	(↑)	**0.904**	**0.893**	**0.826**	**0.909**	**0.842**	**0.725**	**0.935**	**0.900**	**0.833**	**0.990**	**0.978**	**0.895**

for evaluation. The input window size T is set to 20, and the horizon h is set to {3,5,10} in turn. All the parameters of models are trained using the Adam optimizer with weight decay 5e-4, and the dropout rate is set to 0.2. We performed early stopping according to the loss on the validation set to avoid overfitting. The learning rate is chosen from {0.01,0.005,0.001}.

Hyperparameters setting. The hidden dimension of LSTM D and attention layer A is chosen from {16,32,64}. The number of LSTM layers L is chosen from {1,2}. The number of kernels K is chosen from {4,8,12,16}. The output dimension of Adaptive Max Pooling P is chosen from {1,3,5}. The look-back window of AR component q is chosen from {0,10,20}.

4.4 Main Results

We evaluate our model in short-term (horizon = 3) and long-term (horizon = 5,10) settings. Table 2 summarizes the results of all methods. The large difference in RMSE values across different datasets is due to the scale and variance of the datasets, i.e., the scale of the Japan-Prefectures and Canada-Covid is greater than the US-Regions and US-States datasets, which is closely related to the prevalence of epidemics and population density. There is an overall trend that the prediction accuracy drops as the prediction horizon increases because the larger the horizon, the harder the problem.

We observe that the proposed SEFNet achieves the state-of-the-art results on most of the tasks. Traditional statistics methods (AR and LRidge) do not perform well in influenza-related datasets. The main reason is that they are based on oversimplified assumptions and only rely on historical records, they cannot model the strong seasonal effects in influenza datasets. For deep learning-based methods, the performance is improved since they make efforts to deal with nonlinear characteristics and complex patterns behind time series, However, some deep learning-based models work well on some datasets, while not well on others.

(1) **Performance.** The methods mainly focused on dependencies between regions/time series (Cola-GNN and TPA-LSTM) have better performance than the method mainly focused on dependencies between time points (LSTNet), which can point out that inter-series dependencies are quite valuable information. Our proposed model takes inter-series correlations between different regions and temporal relationships in a single region into consideration and carefully fuses them for prediction to achieve better performance.
(2) **Stability.** For Japan-Prefectures and Canada-Covid datasets, the prediction performance of many compared methods greatly decreases when the horizon increases. Because as the variance of the dataset increases, the fluctuation within time series and dependencies between regions are more intricate. SEFNet can make full use of dependencies information, so its predict error rises smoothly and slowly within the growing prediction horizon. Therefore, SEFNet has a better stability.

Through this experiment, it can be concluded that SEFNet has better performance and stability in epidemic forecasting, especially in long-term prediction.

4.5 Ablation Study

In order to clearly verify that the above improvement comes from each added component, we conduct an ablation study on the US-Regions and US-States datasets. Specifically, we remove each component one at a time in SEFNet. We name the model without different components as follows:

- **w/oInter** The model without Inter-Series Embedding module.
- **w/oIntra** The model without Intra-Series Embedding module.
- **w/oAR** The model without the AR component.
- **w/oRAConv** The model uses 1×3 convolution blocks only instead of Region-Aware Convolution.

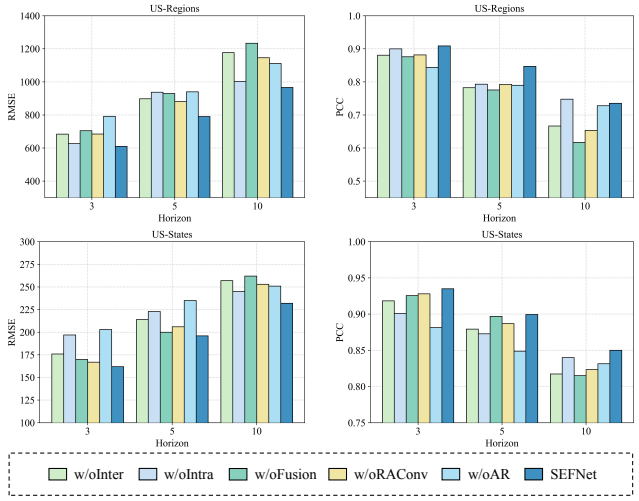


Figure 5: Results of the ablation studies on the US-Region (top) and US-States (bottom) datasets.

- **w/oFusion** The model concatenates Inter-Series Embedding and Intra-Series Embedding directly instead of using Parametric Matrix Fusion.

The ablation results are shown in Figure 5. We highlight several observations from these results:

(1) The full SEFNet model achieves almost the best results.
(2) Directly concatenating two feature embeddings will bring performance drops, while the fusion method used in SEFNet can bring performance gains, especially in long-term prediction (horizon=5,10). Because when the horizon increases, the difficulty of prediction will increase accordingly, and the complex dependencies among regions are more difficult to detect. Parametric-matrix fusion adaptively learns the importance of inter- and intra-series embeddings, which facilitates capturing complex and potential relationships in long-term prediction.
(3) Compared with single-scale convolutions, Region-Aware Convolution brings performance improvement by capturing and aggregating multi-scale features, which proves its strong feature representation power.
(4) Removing the AR component from the full model caused significant performance drops, showing the crucial role of the AR component in general.

This ablation study concludes that our model design is the most robust across all baselines, especially with large horizons.

5 CONCLUSION

In this paper, we propose an Inter- and Intra-**S**eries **E**mbeddings **F**usion **Net**work (SEFNet) for epidemic forecasting. We first extract inter- and intra-series embeddings from two parallel modules. Specifically, in inter-series embedding module, we design a Region-Aware Convolution component that is better to extract feature representations of time series and capture the dynamic dependencies among regions. Then, we fuse two embeddings through parametric-matrix fusion for prediction. To further enhance the robustness, we apply an AutoRegressive component as the linear part. Experiments on four real-world epidemic-related datasets show the proposed model outperforms the state-of-the-art baselines in terms of performance and stability. In future work, we plan to delve into the dynamic dependencies and mutual influences among regions.

ACKNOWLEDGMENTS

This work is supported by the Key R&D Program of Guangdong Province No.2019B010136003 and the National Natural Science Foundation of China No. 62172428, 61732004, 61732022.

REFERENCES

[1] Yue Cui, Chen Zhu, Guanyu Ye, Ziwei Wang, and Kai Zheng. 2021. Into the Unobservables: A Multi-range Encoder-decoder Framework for COVID-19 Prediction. In *Proc. of CIKM.*
[2] Songgaojun Deng, Shusen Wang, Huzefa Rangwala, Lijing Wang, and Yue Ning. 2020. Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction. In *Proc. of CIKM.*
[3] Shuhui Guo, Fan Fang, Tao Zhou, Wei Zhang, Qiang Guo, Rui Zeng, Xiaohong Chen, Jianguo Liu, and Xin Lu. 2021. Improving Google flu trends for COVID-19 estimates using Weibo posts. *Data Science and Management* (2021).
[4] Tiberiu Harko, Francisco SN Lobo, and MK3197716 Mak. 2014. Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. *Appl. Math. Comput.* (2014).
[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).
[6] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. 2019. DSANet: Dual self-attention network for multivariate time series forecasting. In *Proc. of CIKM.*
[7] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML.*
[8] Jayson S Jia, Xin Lu, Yun Yuan, Ge Xu, Jianmin Jia, and Nicholas A Christakis. 2020. Population flow drives spatio-temporal distribution of COVID-19 in China. *Nature* (2020).
[9] Xiaoyong Jin, Yu-Xiang Wang, and Xifeng Yan. 2021. Inter-series attention model for covid-19 forecasting. In *Proc. of SDM.*
[10] Seungwon Jung, Jaeuk Moon, Sungwoo Park, and Eenjun Hwang. 2021. Self-attention-based Deep Learning Network for Regional Influenza Forecasting. *IEEE Journal of Biomedical and Health Informatics* (2021).
[11] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *Proc. of SIGIR.*
[12] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. 2022. Deep Graph Clustering via Dual Correlation Reduction. In *Proc. of AAAI.*
[13] Edson Zangiacomi Martinez, Elisângela Aparecida Soares da Silva, and Amaury Lelis Dal Fabbro. 2011. A SARIMA forecasting model to predict the number of cases of dengue in Campinas, State of São Paulo, Brazil. *Revista da Sociedade Brasileira de Medicina Tropical* (2011).
[14] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* (2019).
[15] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2019. The performance of LSTM and BiLSTM in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data).*
[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proc. of CVPR.*
[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proc. of NeurIPS* (2017).
[18] Zheng Wang, Prithwish Chakraborty, Sumiko R Mekaru, John S Brownstein, Jieping Ye, and Naren Ramakrishnan. 2015. Dynamic poisson autoregression for influenza-like-illness case count prediction. In *Proc. of KDD.*
[19] Miguel Won, Manuel Marques-Pita, Carlota Louro, and Joana Gonçalves-Sá. 2017. Early and real-time detection of seasonal influenza onset. *PLoS computational biology* (2017).
[20] Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. 2018. Deep learning for epidemiological predictions. In *Proc. of SIGIR.*
[21] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proc. of KDD.*
[22] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proc. of AAAI.*

Unsupervised Structure Confidence Sampling for Image Inpainting

Xinrong Hu[1,2], Tao Wang[1,2], Jinxing Liang[1,2]*, Junjie Jin[1,2], Junping Liu[1,2], Tao Peng[1,2], Yuanjun Xia[1,2]
1. Engineering Research Center of Hubei Province for Clothing Information
2.School of Computer Science & Artificial Intelligence, Wuhan Textile University
E-mail: hxr@wtu.edu.cn, wtadota@163.com, jxliang@wtu.edu.cn,junjie.jin@qq.com, ljp@wtu.edu.cn, pt@wtu.edu.cn, xiaujun@163.com

***Abstract*—Context: Current image inpainting methods show great effects in different applications such as image editing, object removal, art creation and soon, but lack of editability of the inpainting results and convincing unsupervised features. Objective: To improve the existing methods, an optimized framework for image inpainting purpose is proposed based on hierarchical variational auto-encoder (VAE) as well as some optimization strategies. Method: Firstly, the VAE is used to extract the distribution of the features of the masked image in different scales, however, it will cause the distribution offset of extracted features which is unfavorable for image inpainting. Therefore, an optimal strategy that sampling the effective feature and invalid feature separately to avoid the offset of feature distribution of the masked image is integrated into the framework. To further improve the formulation of the proposed framework, the same encoder is used to realize the conversion from two domains to the same domain, which is a benefit to enhance the extraction of effective feature regions. In addition, we also introduce the cycle consistency constraints and GAN constraints into the framework to supervise the inpainting process. Result: Experimental results on the available image dataset demonstrate the effectiveness and superiority of the proposed framework.**

Keywords- Image inpainting; Auto-encoder; Self-supervision;

I. INTRODUCTION

Image inpainting is always a fundamental challenge in the field of computer vision. The key problem of image inpainting is how to ensure the integrity and consistency of the filled area to the adjacent, including the content, color attribute as well as tone of the image. Methods that producing incomplete filled effects or artificial effects between filled area and surrounding area are no good ones. Image inpainting has been widely used in many fields such as image editing, object removal, art creation and other tasks [1-5]. However, most of the existing methods that rely on GAN-based [11] image generation of which lose the adaptation to a variety of applications. Fortunately, in recent years VAE-based [16] image generation strategy based on probability introduce the new path to the current researches on image inpainting.

Previously, the GAN-based image inpainting can be divided into the following two categories: the one-stage inpainting method and the progressive method. For the one-stage method [6-10], its hypothesis that all globally valid image information can be obtained at once for image reconstruction. Although they can ensure the consistency of generated information and context semantics, these methods suffer from the problem of pixel discontinuity and semantic gap of the inpainting result, which can be found in the presence of many missing regions [17]. The reason comes from the large pixel difference between the known and missing regions, which leads to a weak correlation and further produces the hole regions.

Different from utilizing prior available features from input, the progressive methods [23, 25-28] consider that missing regions can not be filled completion at once. Therefore, these kinds of methods gradually reasoning feature value in holes region until the missing region is filled. However, these methods fail to consider the difference and correlation between filled areas and some other certain regions. Most importantly, because the image inpainting is iteratively conducted at the image level, the computational cost is very expensive. These kinds of methods always need more efficient computing environment.

Significantly different from the GAN-based method, currently the VAE-based image generation methods [12-15] can be able to generate novel and diverse image samples by mapping the noise of normal distribution to the image. However, if without some optimization and improvement, these methods cannot be directly used for inpainting of diversified scene images, the reasons are listed as follows. Firstly, when applied these kinds of method to inpainting of diversified scene images, the condition label is the masked image itself and there are no paired training images in the training dataset for each condition label. It will lead to there are no conditional training datasets that can explicitly express the condition distribution for the diversity masked images. Secondly, there are strong constraints for inpainting of diversified scene images, which means that the repaired images should keep integrity and consistency in color and texture with the masked image, therefore, it is more vulnerable to suffer from mode collapse than typical image generation.

Based on the limitations of the currents methods described above, we propose an unsupervised image inpainting framework in this paper based on NVAE [14]. The proposed framework relies on the assumption of implicit space sharing of the three domains and is based on domain transformation and differentiated sampling for finer generation effects. For the original NVAE, the feature information decoded by the upper sampling is more complete, the sampling points of the lower sampling will become unavailable due to the presence of the holes. Different from the original NVAE, the proposed inpainting framework firstly fill the hole area and then combine and derive the posterior distribution based on the feature matching strategy. After that, we realize the sharing of the same encoder within two of the domains in the form of

* Corresponding author.
DOI reference number: 10.18293/SEKE2022-026

additional weights since the existence of ternary domains brings too much coding space. At last, the patch discriminator is used to guide image generation to refine image texture. The main innovations of the proposed framework can be summarized as follows.

The sharing of the same encoder has been realized in different domains through weighting the encoder in the proposed framework.

Different sampling methods has been performed on holes region and mask image respectively based on differentiated sampling.

The GAN constraints has been used to refine the image and generate texture.

The remainder of this paper is organized as follows. Section II analyzes the related works, Section III reports our research methodology and loss parameters, Section IV provides our experimental procedure and results, Finally, Section V concludes this study.

II. RELATED WORK

A. One-stage inpainting

Context encoder [6] is firstly introduced into image inpainting for learning semantic content. Global and local [7] discriminators are commonly used to distinguish generated image in global and local regions while enforcing the consistency of generated image in missing regions. Yu et al. [9] firstly introduce patch match in deep feature for filling missing holes. Liu et al. [22] devise a partial convolution to express different weights for different holes region. Liu et al. [17] design a strategy to limit the hole filling characteristics and the relationship between adjacent and outer regions. By introducing contour constraint, Nazeri et al. [23] propose that contour repair can be carried out gradually to fill the global region. Yu et al. [24] normalize pixels from the corrupted and uncorrupted regions separately based on the original inpainting mask to solve the mean and variance shift problem. These methods have improved the image inpainting accuracy somehow but lack effective constraints on the hole center and lack effective semantic reasoning ability in some complex scenarios.

B. Progressive inpainting

Li et al. [25] propose to leverage a shared module to gradually repair the edge of the hole to enhance the constraint on the center of the hole. Yang et al. [26] devise a pyramid structure loss to supervise structure learning and embedding for additional structural constraints. Yi et al. [27] design a contextual residual aggregation module as the residuals of generated features so that the incremental generation of target features ensures the detailed texture of generated results. Zeng et al. [28] propose a deep generation model with a feedback mechanism, which outputs the feature map as well as the result of the repair feature. In their method, the highly trusted feature pixel will be used as the valid information in the next iteration. All these methods infer subsequent features by appending the predicted features as prior knowledge, while the repaired features depend on the features of the previously filled area of the hole, the essence of their methods is that the following inferred features come from the initial effective features. This strong correlation makes it impossible to decouple the effective regions from the holes region.

C. VAE-based inpainting

Zhao et al. [18] use an effective reference image as the image inpainting style and further couple the hole image with the reference image based on cross-attention. Peng et al. [19] develop a structural attention module based on a hierarchical vectorized variational auto-encoder to capture the distance relationship, which allows for a variety of repair results. Wan et al. [20] train two variational auto-encoders to transform old photos and clean photos into two latent spaces, respectively. And the translation between these two latent spaces is learned with synthetic paired data. This method generalizes well to real photos because the domain gap is closed in the compact latent space. Du et al. [21] introduce discrete disentangling representation and adversarial domain adaption into general domain transfer framework, aided by extra self-supervised modules including background and semantic consistency constraints, learning robust representation under dual-domain constraints (for example the feature and image domains).

The goal of VAEs is to train a generative model in the form of $p(x,z)=p(z)p(x\mid z)$ where $p(z)$ is a prior distribution over latent variables z and $p(x\mid z)$ is the likelihood function or decoder that generates data X given variable z. Since the true posterior distribution $p(z\mid x)$ is in general intractable, the generative model is trained with the aid of an approximate posterior distribution or encoder $q(z\mid x)$.

In deep hierarchical VAEs, to increase the expressiveness of both the approximate posterior and prior, the latent variables are partitioned into disjoint groups, z={z1, z2,...,zl}, where L is the number of groups. Then, the prior is presented by $p(z)=\prod_l p(z_l\mid z_{<l})$ and the approximate posterior by $q(z\mid x)=\prod_l q(z_l\mid z_{<l},x)$ where each conditional in the prior $p(z_l\mid z_{<l})$ and the approximate posterior $q(z_l\mid z_{<l},x)$ are represented by factorial Normal distributions. We can write the variational lower bound $L_{vae}(x)$ on $\log(p(x))$ as :

$$L_{vae}(x)=E_{q(z|x)}[\log\ p(x\mid z)]-\sum_l KL(q(z_l\mid x,z_{<l})\parallel p(z_l\mid z_{<l})) \quad (1)$$

The objective is trained using the reparameterization trick.

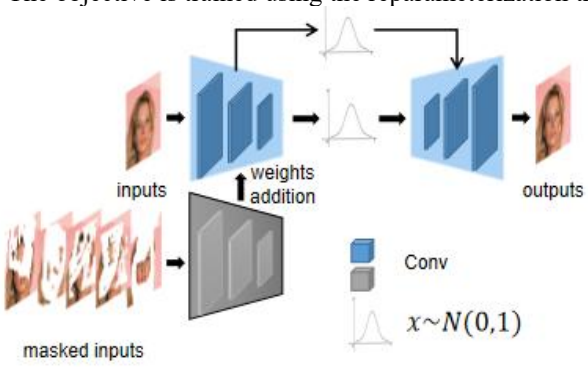


Figure 1. pipeline for image inpainting. The two domains are resolved into a normally distributed space by the same weighted encoder. Multi-level point sampling is used to obtain the distribution relations at different levels, which is conducive to the reliability of the results.

III. PROPOSED FRAMEWORK

For the proposed framework of image inpainting, We firstly introduce assumptions in section A. Then, we introduce in detail the cycle-consistency constraint introduced in the framework in section B. The implementation of the conversion on different domains through a common encoder is described in section C. In section D, the differential sampling method is devised to avoid ill-posed sampling. In section E, various loss functions are proposed, and numerical values are obtained by calculation. Finally, we illustrate in detail the composition of the loss function of the proposed image inpainting framework.

A. Assumption of the proposed framework

Let χ_t and $\chi_m(m=1,2,...)$ be true image domain and m-th mask image domain respectively. In unsupervised image-to-image translation, we are given samples drawn from the marginal distribution $P_{\chi_t(x_t)}$ and $P_{\chi_m(x_m)}$. The goal is to formulate a mapping from p1 to p2 to fill the image with holes. Since an infinite set of possible joint distributions can yield the given marginal distributions, we could infer nothing about the joint distribution from the marginal samples without additional assumptions [14].

Just as the experts can associate the masked image with the true image and consider them as the same image, we hypothesize that they have the shared-latent space. Explicitly, we formulate for any given pair of images x_t and $x_m(m=1,2,...)$, there exists a shared latent code z in a shared latent space, such that we can recover both images from this code, and we can compute this code from each of the two images. That is we postulate there exists functions E, G such that, given a pair of corresponding images (x_t, x_m) from the joint distribution, we have $z = E_t^*(x_t) = E_m^*(x_m)$ and conversely $x_t = G^*(z)$. In this model the function $x_t = F_{m->t}^*(x_m)$ that maps from $F_{m->t}^*(x_m) = G^*(E_m^*(x_m))$, which is a many-to-one mapping. Therefore, we can reconstruct the input image by translating it back to the translated input image. In other words, the proposed shared latent space assumption supports the cycle-consistency assumption, but not vice versa.

B. Self-cycle consistency

Feature learning of missing images deviates from the real image distribution is an important cause of learning failure, and the learning of real valid features will fix the distribution ambiguity caused by this ill deviation. Our goal is to focus on learning the best $F_{m->t}^*$ .The migration feature of ill-conditioned x_m will make the adaptive relationships within the model become locally valid. In the process of learning and training, the local adaptation relationship is further enhanced, and the encoder can't learn the distribution scheme that follows the whole real sample. This is fatal for re-parameterized sampling prior to decoding due to the offset of the distribution. Therefore, we can apply self-cycle consistency constraints in the proposed framework to further regularize the ill-posed unsupervised image inpainting problem. Formally, $x_t := G^*(E_t^*(x_t))$. This ensures complete representation learning within the image range.

C. Encoder-sharing

Based on the assumption of the shared implicit space proposed above, the true image and the masked image are distributed in different domains, so it is a good scheme to use two different encoders to map different domains. However, it notes that because two different encoders learn different spatial features, it does not transfer the learned mapping relationship to the masked image. Correspondingly, we use the same encoder to realize the transformation of two different domains to the same domain through the dynamic weighting method. Formally, we specify the input feature as X and the mask as M. we can get $E_m^* := M \otimes E_t^*$. Similarly, the mask is updated by the following rules:

$$m' = \begin{cases} 1, if\ (sum(X \otimes M)) > 0 \\ 0, otherwise \end{cases} \tag{2}$$

To implement the shared latent space assumption, we further assume a shared intermediate representation h such that the process of generating a pair of corresponding images admits a form of.

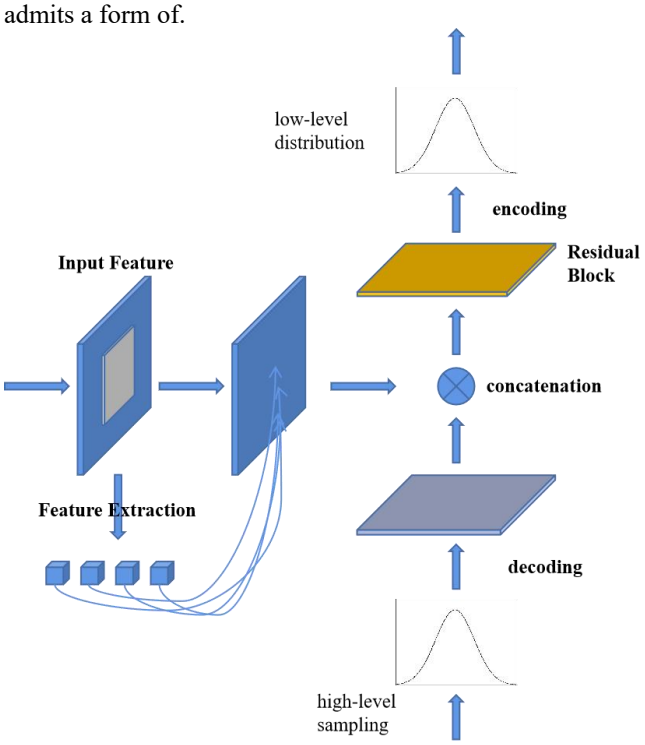


Figure 2. sampling module. The feature distribution obtained by the encoder is firstly filled with the hole features by the block matching strategy. Secondly, the decoded features from the high-level distribution sampling are concatenated to the repaired features to further obtain the joint feature distribution.

D. Distinctive Sampling

The absence of content leads to the ill-conditioned representation of true distribution features, and this ill-conditioned representation further limits sampling errors

due to global content sampling. Correspondingly, the sample errors further lead to the blurring and artificial imprinting of inpainting areas. The fundamental reason for this phenomenon is the use of consistent feature re-parameterized sampling for both valid and invalid feature regions, while ignoring the undesirable bias of invalid feature regions. Therefore, we conduct differentiated sampling in this region, which will no longer sample the features as a whole alone, but focus on the inpainting of sample bias in invalid feature regions. Moreover, it can not be ignored that the characteristics of invalid areas of different sizes and locations correspond to the different distribution of environmental characteristics, so we perform the sampling based on the joint distribution of the environment, regardless of the size and location of the region of the holes. Actually, we formulate an equation $f^i = q(z_i \mid x)$. To obtain the feature distribution of the hole region, we match the similarity of the patch region.

$$sim^i_{x,y,x',y'} = < \frac{f^i_{x,y}}{\| f^i_{x,y} \|}, \frac{f^i_{x',y'}}{\| f^i_{x',y'} \|} > \quad (3)$$

where $sim^i_{x,y,x',y'}$ indicates the similarity between the feature at the location of (x, y) and (x', y') . Further, the softmax function is used to calculate the attention score $score^i_{x,y,x',y'} = soft\max(sim^i_{x,y,x',y'})$. Eventually, attention scores are used to reconstruct the feature distribution.

$$\hat{f}^i_{x,y} = \sum score^i_{x,y,x',y'} f^i_{x,y} \quad (4)$$

The results of the filled feature distribution are linked to the prior distribution from the upper-level, and the joint probability distribution $q(z_i \mid x, z_{i-1})$ is obtained from the existing residual blocks.

E.Loss Function

The loss function of the proposed framework for image inpainting includes the sum of the three sub-loss functions, the KL divergence loss, the reconstruction lass as well as the GAN constraints. Details of the loss functions are described as follows.

1) KL Divergence Loss

Inspired by NAVE hierarchical sampling, we adopt a similar strategy for deeper-levels sampling. Practically, we examine the KL term in L_{vae} , as illustrated in equation

$$\mathrm{KL}(q(z_i|x)||p(z_i)) = \frac{1}{2}\left(\frac{\Delta\mu_i^2}{\sigma_i^2} + \Delta\sigma_i^2 - log\Delta\sigma_i^2 - 1\right) \quad (5)$$

where $\Delta\mu_i$ and $\Delta\sigma_i$ are the relative location and scale of the approximate posterior concerning the prior

2) Reconstruction Loss

Our network translates instance images into completion images in an unsupervised way. At times, the instance image is different from the corresponding completion image in pixel level. It is desired that the instance image is the same as the corresponding completion image in low-dimensional visible space. Therefore, the latent space loss is defined as:

$$L^Z_{rec} = ||E^*_t(x_t) - E^*_t\left(G\left(E^*_m(x_m)\right)\right)||_1 \quad (6)$$

For each masked image x_m there is only one ground true image x_t corresponding to it. When its corresponding ground truth image x_t is used as the guided instance image, the output of the generation module is x_t . Therefore, an identical reconstruction constraint is needed, which is defined as follows:

$$L^g_{rec} = ||x_t - G(E^*_m(x_m))||_1 \quad (7)$$

3) GAN Constraints

To refine our generation effects, we use a patch discriminator to activation the mapping of different range. The adversarial loss is defined as:

$$L_{adv} = \min_{G,E^*_m} \max_D [E_{x_t \sim Pdata} \log D(x_t) + E_{x_m \sim Pdata} \log(1 - D(G(E^*_m(x_m))))] \quad (8)$$

4) Total Loss

As is shown in equation (9), the total loss L_{total} of our method consists of three groups of component losses.

$$L_{total} = \lambda_{vae} L_{vae} + \lambda_{rec}\left(L^Z_{rec} + L^g_{rec}\right) + \lambda_{adv} L_{adv} \quad (9)$$

IV. EXPERIMENTS

The effectiveness and the superiority of the proposed framework for image inpainting are tested on the CelebA dataset [29]. The CelebA dataset contains more than 180,000 training images of face images. All images are re-sized and cropped to 256×256 for training and testing. Our framework is trained using the Adam [30] optimizer with the batch size of 6. We use the initial learning rate as 1e-4 to train the framework, and the learning rate is fine-tuned with learning rate of 5e-5 and decay rate of 0.02. We compare our method with several state-of-the-art methods based on the above-mentioned dataset. The qualitative comparisons between proposed framework and other methods are presented in Figure 3. Compared to other methods, the proposed framework shows more consistency with the true face effect in general and the result produced by the proposed framework with more detailed texture, but the image with some blur.

In addition, we also perform the quantitative comparison between the proposed framework with the NVAE method. It should be note that since the first three methods cannot generate diverse results for image inpainting, therefore these methods have only single output in the numerical comparison with the proposed framework, so the quantitative comparison is only performed to the NVAE method. The metrics of the quantitative comparison is summarized in Table 1 for the CelebA dataset. It can be seen that the proposed framework has better image reconstruction accuracy than NVAE for all the tested metrics of SSIM, PSNR, and MEAN l1. More detailed results are presented as follows.

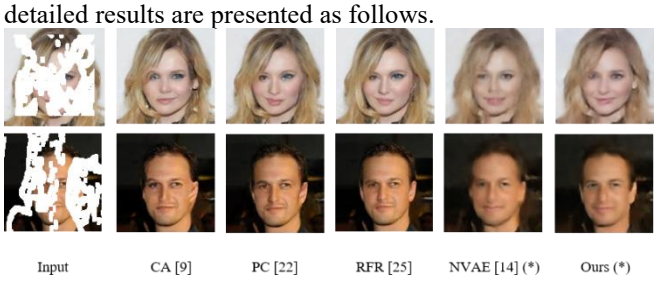


Figure 3. Qualitative comparisons the inpainting effect of the proposed framework (Ours) and existing methods on the CelebA dataset.

A.Diverse Generation

As is reported in section III, since our framework covers multiple levels of sampling, and the sampling parameters can be customized for each level of sampling deviation, of which ensures a variety of results generated. Figure 4 shows the inpainting results when setting different level of sampling parameters. It is showed that under different sampling parameters, the images generated by the guidance have different local features such as mouth shape. Visually compared to the NAVE method, the proposed framework produces more fine textures for the repaired images.

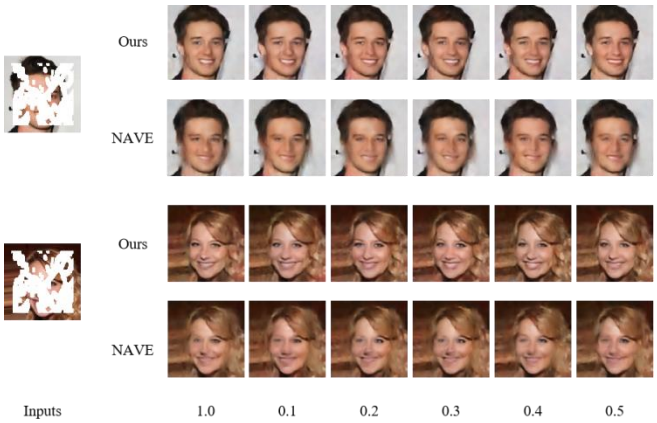


Figure 4. Under different sampling deviations, our method produces better texture characteristics than other method.

	SSIM	PSNR	MEAN l1 (%)
NVAE	0.7055	25.3406	4.0404
Ours	0.7649	27.7054	3.2793

Table 1. Quantitative comparison the inpainting accuracy of the proposed framework (Ours) and NAVE method on CelebA dataset.

B. Effectiveness of Distinctive Sampling

In the proposed framework, we introduce the differentiated sampling module. The differentiated sampling module is essentially a feature repair stack, which is conducive to obtaining more accurate feature distribution information. In order to verify the validity of this module, we test the image inpainting effects with and without the corresponding module in the proposed framework. The comparison is presented in Figure 5. We can see that with the distinctive sampling module we get more pleased inpainting result, where the evaluation metrics of SSIM, PSNR, MEAN L1 get improvement with the distinctive sampling module.

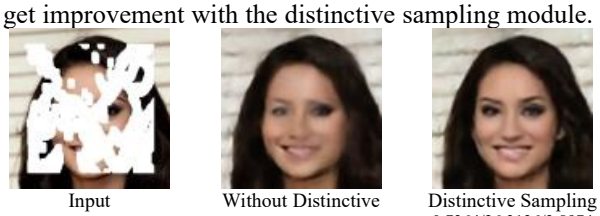


Figure 5. Comparison of image inpainting results with and without the corresponding module in the proposed framework (ssim/psnr/mean l1). The effectiveness of distinctive sampling can be inferred from metrics of SSIM, PSNR and MEAN l1.

V.CONCLUSIONS

In this paper, we propose an unsupervised image inpainting framework, of which can ensures the generation of credible results through domain sharing hypothesis with KL and refactoring constraints. We come up the strategy of encoder sharing to greatly simplifies the proposed framework. In addition, in order to ensure the repaired image have great consistent texture features with ground truth in the hole areas, we also devise a differentiated sampling strategy to distinguish the sampling parameters of the mask region and the effective region. Experimental results show the effectiveness and superiority of the proposed framework, which verifies the hypothesis behind the proposed method. However, there still have some defects for the proposed framework in current stage, for example, the blur phenomenon in the generated images. On one hand, this phenomenon comes from the limitation of VAE-based image generation. On the other hand, some features of the input image are missing, which will inevitably lead to the deviation of the overall features of the image. Although we have filled the missing features in the generation process of the proposed framework, however, this deviation is irreversible. In the next step of our research, we are going to address this issue and further improve the image inpainting effect.

REFERENCES

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. In ACM SIGGRAPH 2009 Papers (New Orleans, Louisiana) (SIGGRAPH '09). Association for Computing Machinery, New York, NY, USA, Article 24, 11 pages.

[2] A. Criminisi, P. Perez, and K. Toyama. 2004. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing 13, 9(2004), 1200–1212.

[3] Zhan, X., Pan, X., Dai, B., Liu, Z., Lin, D., & Loy, C. C. 2020. Self-supervised scene de-occlusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 3784-3792.

[4] Rakshith Shetty, Mario Fritz, and Bernt Schiele. 2018. Adversarial Scene Editing: Automatic Object Removal from Weak Supervision. In Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, 7717–7727.

[5] Linsen Song, Jie Cao, Lingxiao Song, Yibo Hu, and Ran He. 2019. Geometry-aware face completion and editing. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 2506–2513.

[6] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. 2016. Context Encoders: Feature Learning by Inpainting. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2536–2544.

[7] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and Locally Consistent Image Completion. ACM Trans. Graph. 36, 4, Article 107 (July 2017), 14 pages.

[8] Y. Zeng, J. Fu, H. Chao, and B. Guo. 2019. Learning Pyramid-Context Encoder Network for High-Quality Image Inpainting. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1486–1494.

[9] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. 2018. Generative Image Inpainting with Contextual Attention. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 5505–5514.

[10] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang. 2019. Free-Form Image Inpainting With Gated Convolution. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 4470–4479.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative Adversarial Networks. Commun. ACM 63, 11 (Oct. 2020), 139–144.

[12] Neural Discrete Representation Learning.Oord, A. V. D., Vinyals, O., & Kavukcuoglu, K. 2017. Neural discrete representation learning. arXiv preprint arXiv:1711.00937.
[13] Razavi, A., van den Oord, A., & Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. In Advances in neural information processing systems. 14866-14876.
[14] Vahdat, A., & Kautz, J. 2020. Nvae: A deep hierarchical variational autoencoder. arXiv preprint arXiv:2007.03898.
[15] Liu, H., Wan, Z., Huang, W., Song, Y., Han, X., & Liao, J. 2021. PD-GAN: Probabilistic Diverse GAN for Image Inpainting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 9371-9381.
[16] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
[17] H. Liu, B. Jiang, Y. Xiao, and C. Yang. 2019. Coherent Semantic Attention for Image Inpainting. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 4169–4178.
[18] Zhao, L., Mo, Q., Lin, S., Wang, Z., Zuo, Z., Chen, H., ... & Lu, D. 2020. Uctgan: Diverse image inpainting based on unsupervised cross-space translation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 5741-5750.
[19] Du, W., Chen, H., & Yang, H. 2020. Learning invariant representation for unsupervised image restoration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 14483-14492.
[20] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In Proceedings of the European Conference on Computer Vision (ECCV). 85–100.
[21] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi. 2019. EdgeConnect: Structure Guided Image Inpainting using Edge Prediction. In 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). 3265–3274.
[22] Tao Yu, Zongyu Guo, Xin Jin, Shilin Wu, Zhibo Chen, Weiping Li, Zhizheng Zhang, and Sen Liu. 2020. Region Normalization for Image Inpainting. In AAAI. 12733–12740.
[23] J. Li, N. Wang, L. Zhang, B. Du, and D. Tao. 2020. Recurrent Feature Reasoning forImage Inpainting. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 7757–7765.
[24] Jie Yang, Zhiquan Qi, and Yong Shi. 2020. Learning to incorporate structure knowledge for image inpainting. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 12605–12612.
[25] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu. 2020. Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 7505–7514.
[26] Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. 2020. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In European Conference on Computer Vision. Springer, 1–17.
[27] Z. Liu, P. Luo, X. Wang, and X. Tang. 2015. Deep Learning Face Attributes in the Wild. In 2015 IEEE International Conference on Computer Vision (ICCV). 3730–3738.
[28] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

Verifying BDI Agents in Dynamic Environments

Blair Archibald, Muffy Calder, Michele Sevegnani, Mengwei Xu
School of Computing Science, University of Glasgow, Glasgow, UK
{blair.archibald, muffy.calder, michele.sevegnani, mengwei.xu}@glasgow.ac.uk

***Abstract*—The Belief-Desire-Intention (BDI) architecture is a popular framework for rational agents, yet most verification approaches are limited to analysing the behaviours of an agent in a subset of all possible environments. However, in practice, BDI agents operate in dynamic environments where the exact occurrence of external changes is difficult to predict. For safety/security we need to assess whether the agent behaves as required in all circumstances. To address this, we define environments, accounting for both sensor information about physical changes and new tasks to be completed, as a non-deterministic finite-state automata. We give an environment-enabled extension to the Conceptual Agent Notation (CAN) language including an executable semantics via an encoding to Milner's bigraphs and the BigraphER tool. We illustrate the framework through a simple Unmanned Aerial Vehicle (UAV) example that is verified using mainstream tools including PRISM model checker. Results show our approach can automatically identify agent design flaws to aid agent programmers in design, debugging, and analysis.**

***Index Terms*—BDI Agents, Formal Methods, Environments**

I. Introduction

Belief-Desire-Intention (BDI) is one of the most popular agent development frameworks and forms the basis of many agent-oriented programming languages including AgentSpeak [1], CAN [2], and 3APL [3]. (B)eliefs represent what the agent knows, (D)esires what the agent wants to bring about, and (I)ntentions the desires the agent is currently acting upon. BDI features a collection of mature software and platforms including JACK [4] and Jason [5].

BDI agents rarely stand alone but instead exist in an environment—a source of information that the agent has been designed to understand—and operate by means of a reasoning cycle (see in Fig. 1) with three consecutive steps: (1) *perceive*, (2) *deliberate* and (3) *act*. The first step is to perceive the environment to update the agent's beliefs, the second to deliberate to determine, for example, what plan to select under current beliefs (specified by language semantics), and the third to act on the external environment by executing actions of the selected plans. This approach facilitates practical agents that effectively interact with their environments. However, designing agents that ensure correct behaviours in *all* possible environments is very difficult. As agent systems (*e.g.* robotic systems [6]) are increasingly employed in many real-life (*e.g.* domestic and industrial) settings, we must tackle this problem as it is crucial to analyse agent behaviours under all circumstances.

To illustrate the issues, we consider scenarios from Unmanned Aerial Vehicles (UAVs). In searching operations,

DOI reference number: 10.18293/SEKE2022-149

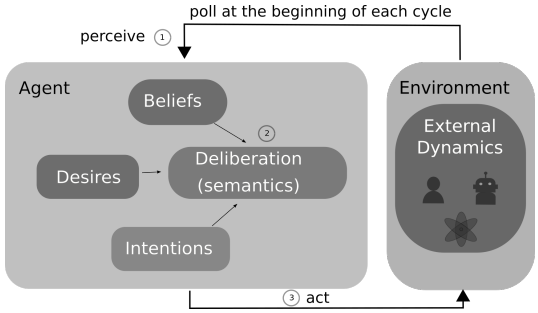


Figure 1: Agent interactions with an environment consisting of external physical changes due to *e.g.* natural phenomena. Numbers give the order of operation. All information is fed into the belief base.

UAVs patrol a designated area to identify objects of interest, *e.g.* missing persons, returning to base when detection is successful. During the mission, UAVs are expected to interact with the physical world. For example, UAVs should activate parking mode if the weather becomes harsh. The changes of these physical attributes (*e.g.* the weather and objects available to be detected) can happen at any time during UAV's mission and all combinations of these changes may produce complex behaviours. Therefore, we need mechanisms (*e.g.* verification techniques), involving an automated exhaustive analysis, in order to assess whether agents will always behave correctly under any environmental conditions.

Traditionally the analysis of agent behaviours is carried out by computational simulations. For example, the BDI platform Jason supports simulated environments that provide certain services to the agent (the ability to perceive and take action). Although computational simulations are quick to run, they are generally non-exhaustive in terms of agent behaviours in a given environment. As such, *actual* agent behaviours in deployment may not be the seen in simulation and rare cases may not manifest despite many simulation runs. To analyse all possible runs of agent behaviours in one environment, formal verification can be applied to build a mathematical model of the agent system [7]. However, if the agents are to be used in safety-critical areas, or where agent mistakes might involve financial penalties, this approach guarantees very little about agent behaviours in *actual* environments (which are difficult to accurately predict at design stage). To provide stronger guarantees, we instead assess *all* possible agent behaviours in *all* possible environments. A graphical comparison between these approaches is in Fig. 2.

We provide a verification framework based on Bi-

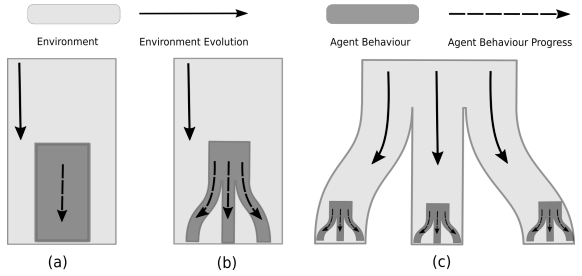


Figure 2: Approaches to analyse agent behaviours in environments. (a) Simulation: one run of agent behaviour in one environment; (b) existing verification approaches: all possible agent behaviours in one environment; (c) Proposed approach: all possible agent behaviours in all possible environments.

graphs [8]—a graph-based universal modelling formalism—that models both BDI agents and environments. We build on previous work [9] on a bigraph encoding of CAN semantics [2] (that includes classical BDI features and advanced features such as declarative goals), that provides an *executable* semantics for BDI agents operating in a (dynamic) environment. Verification is achieved through mainstream software tools including BigraphER and PRISM [10], and demonstrated by verifying several properties of UAVs.

As the external environment *with respect to an agent* may change while the agent is reasoning, the key of our approach is to provide a formalisation of external dynamics that summarises the main environment changes (due to *e.g.* natural phenomena) over each of the reasoning cycle. In other words, how the external environment has been updated in reality over each cycle is subject to the nature of the environment and what matters is that the agent can access the final effects of what has changed over this cycle. By focusing on reasoning cycles, we remain agnostic to the actual time required for each cycle, which can be difficult to anticipate at design stage due to the delay or variation in real process deployed on the hardware. This approach offers a feasible way to specify the dynamics of external environments while enabling practical verification by avoiding both state explosion and difficult time synchronization problem between agent reasoning and environment simulation when using real-time.

We make the following research contributions:

- We formalise the dynamics of external environments due to *e.g.* natural phenomena and other agents as a finite non-deterministic finite-state automata.
- We provide a verification framework for environment-enabled CAN agents via bigraphs that guarantees expected agent behaviours under all possible environments.
- We showcase our formal verification approach and illustrate how it can improve the design of the BDI agents.

The paper is organised as follows. In Section II we review BDI agents and Bigraphs. In Section III we define the framework for agents in dynamic environments. In Section V we demonstrate our approach using a UAV example. We discuss related work in Section VI and conclude in Section VII.

II. BACKGROUND

A. BDI Agents

The CAN language formalises classical BDI agents consisting of a belief base $\mathcal{B}$ and a plan library Π. The belief base $\mathcal{B}$ is a set of formulas encoding the current beliefs from a language $\mathcal{L}$ and has belief operators for entailment (*i.e.* $\mathcal{B} \models \varphi$), and belief atom addition (resp. deletion) $\mathcal{B} \cup \{b\}$ (resp. $\mathcal{B} \setminus \{b\}$)[1]. A plan library Π is a collection of plans of the form $e : \varphi \leftarrow P$ with e the triggering event, φ the context condition, and P the plan-body. Events can be either external or internal (*i.e.* sub-goals that the agent tries to accomplish). We also use E to denote the set of events (i.e. the triggering event) in the plan library. The language used in the plan-body is defined by $P = nil \mid act \mid e \mid P_1; P_2 \mid P_1 \triangleright P_2 \mid P_1 \parallel P_2 \mid e : (|\varphi_1 : P_1, \cdots, \varphi_n : P_n|) \mid goal(\varphi_s, P, \varphi_f)$ with nil an empty program, act a primitive action, and e an internal event. In addition, we use $P_1; P_2$ for sequence, $P_1 \triangleright P_2$ to first try P_1 and use P_2 in case of failure, and $P_1 \parallel P_2$ for interleaved concurrency. A set of relevant plans (those that respond to the same event) is denoted by $e : (|\psi_1 : P_1, \cdots, \psi_n : P_n|)$. A goal program $goal(\varphi_s, P, \varphi_f)$ states that the declarative goal φ_s should be achieved by repeatedly executing P, failing if φ_f holds and exiting successfully if φ_s holds (see [11] for full details). The action act is in the form of $act : \varphi \leftarrow \phi^-; \phi^+$ where φ is a precondition, and ϕ^- and ϕ^+ are the sets of belief atoms to be deleted and added after the action is executed.

CAN semantics is specified by two types of transitions. The first, denoted $\rightarrow$, specifies *intention-level* evolution on configurations $\langle \mathcal{B}, P \rangle$ where $\mathcal{B}$ is the belief base, and P the plan-body currently being executed. The second type, denoted $\Rightarrow$, specifies *agent-level* evolution over $\langle E^e, \mathcal{B}, \Gamma \rangle$, detailing how to execute a complete agent where E^e is the set of pending external events to address (a.k.a. desires), $\mathcal{B}$ the belief base, and Γ a set of partially executed plan-bodies (intentions).

Fig. 3 gives rules for evolving any single intention. For example, the rule act handles the execution of an action, when the pre-condition ψ is met, resulting in a belief state update. Rule $event$ replaces an event with the set of relevant plans, while rule $select$ chooses an applicable plan from a set of relevant plans while retaining un-selected plans as backups. With these backup plans, the rules for failure recovery $\triangleright_;$, $\triangleright_\top$, and $\triangleright_\bot$ enable new plans to be selected if the current plan fails (*e.g.* due to environment changes). Rules ; and $;_\top$ allow executing plan-bodies in sequence, while rules $\|_1$, $\|_2$, and $\|_\top$ specify how to execute (interleaved) concurrent programs. Rules G_s and G_f deal with declarative goals when either the success condition φ_s or the failure condition φ_f become true. Rule G_{init} initialises persistence by setting the program in the declarative goal to be $P \triangleright P$, *i.e.* if P fails try P again, and rule $G_;$ takes care of performing a single step on an already initialised program. Finally, the derivation rule $G_\triangleright$ re-starts the original program if the current program has finished or got blocked (when neither φ_s nor φ_f becomes true).

[1]Any logic is allowed providing entailment is supported. A propositional logic is used in our example.

$$\frac{act : \psi \leftarrow \langle \phi^-, \phi^+ \rangle \quad \mathcal{B} \vDash \psi}{\langle \mathcal{B}, act \rangle \rightarrow \langle (\mathcal{B} \setminus \phi^- \cup \phi^+), nil \rangle}\, act \qquad \frac{\Delta = \{\varphi : P \mid (e' = \varphi \leftarrow P) \in \Pi \wedge e' = e\}}{\langle \mathcal{B}, e \rangle \rightarrow \langle \mathcal{B}, e : (\mid \Delta \mid) \rangle}\, event \qquad \frac{\varphi : P \in \Delta \quad \mathcal{B} \models \varphi}{\langle \mathcal{B}, e : (\mid \Delta \mid) \rangle \rightarrow \langle \mathcal{B}, P \rhd e : (\mid \Delta \setminus \{\varphi : P\} \mid) \rangle}\, select$$

$$\frac{\langle \mathcal{B}, P_1 \rangle \rightarrow \langle \mathcal{B}', P_1' \rangle}{\langle \mathcal{B}, P_1 \rhd P_2 \rangle \rightarrow \langle \mathcal{B}', P_1' \rhd P_2 \rangle}\, \rhd_; \qquad \frac{}{\langle \mathcal{B}, (nil \rhd P_2) \rangle \rightarrow \langle \mathcal{B}', nil \rangle}\, \rhd_\top \qquad \frac{P_1 \neq nil \;\; \langle \mathcal{B}, P_1 \rangle \nrightarrow \;\; \langle \mathcal{B}, P_2 \rangle \rightarrow \langle \mathcal{B}', P_2' \rangle}{\langle \mathcal{B}, P_1 \rhd P_2 \rangle \rightarrow \langle \mathcal{B}', P_2' \rangle}\, \rhd_\bot \qquad \frac{\langle \mathcal{B}, P \rangle \rightarrow \langle \mathcal{B}', P' \rangle}{\langle \mathcal{B}, (nil; P) \rangle \rightarrow \langle \mathcal{B}', P' \rangle}\, ;_\top$$

$$\frac{\langle \mathcal{B}, P_1 \rangle \rightarrow \langle \mathcal{B}', P_1' \rangle}{\langle \mathcal{B}, (P_1; P_2) \rangle \rightarrow \langle \mathcal{B}', (P_1'; P_2) \rangle}\, ; \qquad \frac{\langle \mathcal{B}, P_1 \rangle \rightarrow \langle \mathcal{B}', P_1' \rangle}{\langle \mathcal{B}, (P_1 \| P_2) \rangle \rightarrow \langle \mathcal{B}', (P_1' \| P_2) \rangle}\, \|_1 \qquad \frac{\langle \mathcal{B}, P_2 \rangle \rightarrow \langle \mathcal{B}', P_2' \rangle}{\langle \mathcal{B}, (P_1 \| P_2) \rangle \rightarrow \langle \mathcal{B}', (P_1 \| P_2') \rangle}\, \|_2 \qquad \frac{}{\langle \mathcal{B}, (nil \| nil) \rangle \rightarrow \langle \mathcal{B}, nil \rangle}\, \|_\top$$

$$\frac{\mathcal{B} \models \varphi_s}{\langle \mathcal{B}, goal(\varphi_s, P, \varphi_f) \rangle \rightarrow \langle \mathcal{B}, nil \rangle}\, G_s \qquad \frac{\mathcal{B} \models \varphi_f}{\langle \mathcal{B}, goal(\varphi_s, P, \varphi_f) \rangle \rightarrow \langle \mathcal{B}, ?false \rangle}\, G_f \qquad \frac{P \neq P_1 \rhd P_2 \quad \mathcal{B} \not\models \varphi_s \quad \mathcal{B} \not\models \varphi_f}{\langle \mathcal{B}, goal(\varphi_s, P, \varphi_f) \rangle \rightarrow \langle \mathcal{B}, goal(\varphi_s, P \rhd P, \varphi_f) \rangle}\, G_{init}$$

$$\frac{\mathcal{B} \not\models \varphi_s \quad \mathcal{B} \not\models \varphi_f \quad \langle \mathcal{B}, P_1 \rangle \rightarrow \langle \mathcal{B}', P_1' \rangle}{\langle \mathcal{B}, goal(\varphi_s, P_1 \rhd P_2, \varphi_f) \rangle \rightarrow \langle \mathcal{B}', goal(\varphi_s, P_1' \rhd P_2, \varphi_f) \rangle}\, G_; \qquad \frac{\mathcal{B} \not\models \varphi_s \quad \mathcal{B} \not\models \varphi_f \quad \langle \mathcal{B}, P_1 \rangle \nrightarrow}{\langle \mathcal{B}, goal(\varphi_s, P_1 \rhd P_2, \varphi_f) \rangle \rightarrow \langle \mathcal{B}, goal(\varphi_s, P_2 \rhd P_2, \varphi_f) \rangle}\, G_\rhd$$

Figure 3: Intention-level CAN semantics.

$$\frac{e \in E^e}{\langle E^e, \mathcal{B}, \Gamma \rangle \Rightarrow \langle E^e \setminus \{e\}, \mathcal{B}, \Gamma \cup \{e\} \rangle}\, A_{event} \qquad \frac{P \in \Gamma \quad \langle \mathcal{B}, P \rangle \rightarrow \langle \mathcal{B}', P' \rangle}{\langle E^e, \mathcal{B}, \Gamma \rangle \Rightarrow \langle E^e, \mathcal{B}', (\Gamma \setminus \{P\}) \cup \{P'\} \rangle}\, A_{step} \qquad \frac{P \in \Gamma \quad \langle \mathcal{B}, P \rangle \nrightarrow}{\langle E^e, \mathcal{B}, \Gamma \rangle \Rightarrow \langle E^e, \mathcal{B}, \Gamma \setminus \{P\} \rangle}\, A_{update}$$

Figure 4: Agent-level CAN semantics.

The agent-level semantics are given in Fig. 4. The rule A_{event} handles external events, that originate from the environment, by adopting them as intentions. Rule A_{step} selects an intention from the intention base, and evolves a single step w.r.t. the intention-level transition, while A_{update} discards any unprogressable intentions (either already succeeded, or failed).

B. Bigraphs

Bigraphs are a graph-based universal modelling formalism, introduced by Milner [8] for describing systems with both spatial relationships and non-local linking. They have been used for modelling ubiquitous systems including [12], [13], and as a unifying theory of process calculi [14]. Bigraph dynamics are described using a rewriting system specified via reaction rule $l \longrightarrow\!\!\!\!\blacktriangleright r$ that replace a bigraph matching l (in some larger bigraphs) with a bigraph matching r. Given an initial bigraph and set of reaction rules we derive a (non-deterministic) transition system capturing the behaviour of the system through repeated rule application.

We have used bigraphs to encode the existing CAN language semantics to symbolically analyse BDI agent behaviour [9]. The encoding defines an equivalent bigraph for any CAN agent, and defines reaction rules that faithfully model the operational semantics. To execute bigraphical reactive systems, we use BigraphER [15], an open-source language and toolkit for bigraphs. BigraphER allows exporting transitions systems, *e.g.* DTMCs, for analysis in PRISM. To aid writing logical formulas over transition systems, states are labelled using bigraph patterns that assigns a state a *predicate* label if it contains (a match of) a given bigraph pattern.

III. FRAMEWORK

In this section, we first formalise our notion of an environment and then extend the existing CAN semantics to enable the agent to perceive and alter the environment.

A. Environments

Most existing BDI frameworks either omit the environment (*e.g.* CAN) or provide it with an informal treatment (*e.g.* AgentSpeak). In order to analyse an agent behaviour in an environment we first formalise environments. Following [16], we consider environments accounting for both sensor information (a set of beliefs literals) and external events (a set of tasks to complete). Recall the formula in a belief base of a BDI agent is from the language $\mathcal{L}$ and the set of events the agent can respond to is E. The alphabet for an environment is then $\mathcal{E} = \mathcal{L} \cup E$. An environment state, representing the environment at a point in time, is defined as follows:

Definition 1. *An environment state is $\Theta \in Q$ such that $Q = 2^{\mathcal{E}}$ and for any formula b from language $\mathcal{L}$, if $b \in \Theta$, then $\neg b \notin \Theta$.*

Environment states are sets of belief formulae and newly requested events which are held in the environment at some point. We assume an underlying mechanism that converts from real-world environments to symbolic literals, *e.g.* through pattern detection from sensor input. The condition ensures that no environment states has information indicating that both b and $\neg b$ are true, *i.e.* the law of the excluded middle.

Example 1. *A UAV that perceives an environment state $\Theta = \{\neg\mathsf{harsh\_weather}\}$ believes (based on sensor information) that the weather is not harsh.*

To support cases when an agent changes the environment but the effects of acting are undone, *e.g.* by humans, before the agent starts perceiving in the next cycle, we separate environment changes imposed by the agent from environment changes *e.g.* due to external factors. Environment dynamics contributed by the agent will be formalised through the extensions of CAN semantics in Section III-B. Meanwhile, we give the following external dynamics *with respect to an agent* for an environment over each agent reasoning cycle.

Definition 2. *The external dynamics for an agent in an environment is a tuple $\langle Q, \Theta_0, \epsilon \rangle$ where $Q = 2^{\mathcal{E}}$ is a set of environment states, and $\Theta_0 \in Q$ an initial (or start) state, and $\delta : Q \rightarrow 2^Q$ a finite nondeterministic function transitioning a state $\Theta \in Q$ to a finite set of successor states $\{\Theta_1, \ldots, \Theta_n\}$.*

Essentially, external dynamics of an environment is formalised as a non-deterministic finite-state automata and it, by definition, is *relative* to the agent under consideration, as any other agent is a part of the external environment.[2] Although we assume the automata to be finite-state, this does not preclude infinite behaviour, *e.g.* looping between existing states indefinitely.

Example 2. *Following Example 1, we assume the current environment state* $\Theta = \{\neg\mathsf{harsh\_weather}\}$. *Then a possible finite set of successor states can be* $\Theta_1 = \{\neg\mathsf{harsh\_weather}\}$, $\Theta_2 = \{\neg\mathsf{harsh\_weather}, \mathsf{e\_deliver}\}$, $\Theta_3 = \{\mathsf{harsh\_weather}\}$ $\Theta_4 = \{\mathsf{harsh\_weather}, \mathsf{e\_deliver}\}$ *where* e_deliver *is a new (external) event. These four states describes all combinations of physical attribute* harsh_weather *and the event* e_deliver[3].

In the following section, we extend CAN semantics to interact with the external changes in an environment.

B. Perceiving and Acting

Currently CAN does not support explicit environments. To allow environments, we augment both intention level configuration $\langle \mathcal{B}, P\rangle$ (resp. agent-level configuration) in CAN with an environment state Θ, namely $[\langle \mathcal{B}, P\rangle, \Theta]$ (resp. $[\langle E^e, \mathcal{B}, \Gamma\rangle, \Theta]$) where $\mathcal{B}$ is the belief base, P the current intention, E^e the external event set, and Γ the set of intentions.

When perceiving, the belief base $\mathcal{B}$ of an agent is updated to reflect[4] the current environment state Θ where $\mathcal{B} = \Theta$. While sensed information may remain in the environment, the perceived new events should be deleted to avoid them being adopted twice, *i.e.* these are explicitly removed from the environment. The following rule $A_{perceive}$ is given.

$$\frac{}{[\langle E^e, \mathcal{B}, \Gamma\rangle, \Theta] \Rightarrow [\langle E^e, \Theta, \Gamma\rangle, \Theta \setminus \{e \mid e \in \Theta\}]} \; A_{perceive}$$

After perceiving, the agent assimilates perceived new events to the external event set at some later reasoning cycle and subsequently removed from the belief base (rule $A_{assimilate}$).

$$\frac{e \in \mathcal{B}}{[\langle E^e, \mathcal{B}, \Gamma\rangle, \Theta] \Rightarrow [\langle E^e \cup \{e\}, \mathcal{B} \setminus \{e\}, \Gamma\rangle, \Theta]} \; A_{assimilate}$$

Alongside external dynamics, the agent can alter the environment through acting. Although there is already an intention-level rule in CAN for acting, it only changes the belief base (rule act in Fig. 3) which assumes that the agent immediately believes the action has had an effect on the environment. However as, for example, the actions on an agent might be undone another agent, an agent should wait for the effects to take place in the environment and update its beliefs only *after* sensing from the environment in the next reasoning cycle (through rule $A_{perceive}$). As such, we replace the old act rule with the following rule act^{new} to apply effects of actions always in the *environment*, not the belief base.

$$\frac{act : \psi \leftarrow \langle \phi^-, \phi^+\rangle \qquad \mathcal{B} \vDash \psi}{[\langle \mathcal{B}, act\rangle, \Theta] \rightarrow [\langle \mathcal{B}, nil\rangle, (\Theta \setminus \phi^-) \cup \phi^+]} \; act^{new}$$

Finally, the update of an environment due to the external dynamics is applied in the beginning of each reasoning cycle according to Definition 2 and is detailed in Section IV

IV. AGENT REASONING CYCLE

We encode the agent cycle (Fig. 1) including external dynamics of an environment, into bigraphs to obtain an executable semantics (available online[5]) that we use for verification. The cycle is a three step process described as follows:

Step 1 (environment update): Apply the non-deterministic function δ in Definition 2 to update the current environment state Θ to a finite set of states $\delta(\Theta) = \{\Theta_1, \ldots, \Theta_n\}$.

Step 2 (perceive): Using rule $A_{perceive}$, an agent perceives everything in the current (newly updated) environment.

Step 3 (progress): Progress the agent using rule $A_{assimilate}$ or rules in Fig. 4. When A^{new}_{step} is applied, it selects an intention non-deterministically from the intention base, and evolves a single step w.r.t. intention-level transition (seen in Fig. 3).

The reasoning cycle is repeated until all external events and intentions are fully addressed and no new tasks are requested. As we seek to examine the agent behaviours, we stop analysis when the agent stops operating.

To analyse an agent in all possible environments, model checking is applied to the transition system generated from our executable semantics. As we use bigraphs, the transition system has bigraphs as states and rewrite rules as transitions. To reason over the transition system, we label states with *bigraph patterns* [13], and specify dynamic properties using linear or branching time temporal logics such as Computation Tree logic (CTL) [17]. As we generate a transition system, the property specification language is constrained by the model checker. Here we use the non-probabilistic and non-reward logics provided by PRISM[6].

V. UAVS EXAMPLE

To illustrate our framework, we consider a small example taken from UAV surveillance mission systems [9].

A. Design

A UAV patrols a pre-defined area to identify objects of interest and can park itself upon harsh weather to avoid damage. The agent design and environment specifications are given in Fig. 5. The first plan (line 2) addresses event e_patrol_init and is always applicable (true context). The play-body consists of a declarative goal, $\mathsf{goal}(\mathsf{detection}, \mathsf{e\_patrol\_task}, \mathsf{false})$, that continually pursues e_patrol_task until an object of interest is detected (*i.e.* detection). No failure condition is specified. If this goal completes, the action return is executed to return to base, the effect is that result in that returned holds (unshown here). The event e_patrol_task is handled by plan on line 3 which contains a further declarative goal $\mathsf{goal}(\mathsf{harsh\_weather} \vee \mathsf{battery\_low}, \mathsf{e\_patrol}, \mathsf{false})$ instructing

[2]This definition is a special case when the agent is fixed and can be extended in future to allow multi-agents in a shared environment.

[3]Unlike belief formulae built from a language, the negation of an event in an environment is denoted as the absence of such an event.

[4]Partial observability is not currently supported.

[5]https://bitbucket.org/uog-bigraph/bdi_env_model_seke22/src/master/

[6]As this is supported natively by BigraphER.

1 Plan library
2 e_patrol_init : true ← goal(detection, e_patrol_task, false); return
3 e_patrol_task : true ← goal(harsh_weather, e_patrol, false); e_pause
4 e_patrol : true ← patrol
5 e_pause : harsh_weather ∧ ¬parked ← activate_parking; wait
6 e_pause : harsh_weather ∧ parked ← wait
7 initial environment state
8 $\Theta_0 = \{\neg a, \neg b, \neg c, \neg d, e\_patrol\_init\}$
9 environment transition function

$$\delta(\Theta) = \begin{cases} \{\Theta, (\Theta \setminus \{\neg a\}) \cup \{a\}, (\Theta \setminus \{\neg b\}) \cup \{b\}, (\Theta \setminus \{\neg a, \neg b\}) \cup \{a, b\}\} & \text{if } \neg a \wedge \neg b \in \Theta \quad (1) \\ \{\Theta, (\Theta \setminus \{\neg a\}) \cup \{a\}\}, & \text{if } \neg a \wedge b \in \Theta \quad (2) \\ \{\Theta, (\Theta \setminus \{\neg b\}) \cup \{b\}\}, & \text{if } a \wedge \neg b \in \Theta \quad (3) \\ \{\Theta\}, & \text{if } a \wedge b \in \Theta \quad (4) \\ \{(\Theta \setminus \{b, c\}) \cup \{\neg b, \neg c\}\}, & \text{if } b \wedge c \in \Theta \quad (5) \end{cases}$$

where a = detection, b = harsh_weather, c = waited (the effect of action wait) and d = returned (the effect of action return).

Figure 5: Patrolling Task Design for BDI Agents.

1 Plan library
2 e_patrol_init : true ← goal(¬harsh_weather ∧ detection, e_patrol_task, false); goal(¬harsh_weather ∧ returned, e_return_task, false)
3 e_patrol_task : true ← goal(harsh_weather, e_patrol, false); e_pause
4 e_return_task : true ← goal(harsh_weather, e_return, false); e_pause
5 e_patrol : true ← patrol
6 e_return : true ← return
7 e_pause : harsh_weather ∧ ¬parked ← activate_parking; wait
8 e_pause : harsh_weather ∧ parked ← wait

Figure 6: Discovered Corrections of Fig. 5.

an agent to patrol continuously unless the weather is harsh when it should pause (*i.e.* event e_pause). Plans in lines 5 to 6 handle the e_pause by waiting until weather becomes better, while the plan in line 4 performs the patrolling. For succinctness, descriptions of actions (*i.e.* their precondition and effects) such as return and wait are not shown but can be found in our online model.

To specify environments, we assume favourable conditions (as often in practice) in the initial state (line 8). The environment transition function (line 9) describes the set of successor states given the current states: the object becoming available for detection or harsh weather appearing can happen at any time so long as they have not already occurred (cases 1–3); no update is available if detection and harsh weather are present (case 4); and, for practicality, the harsh weather will always get better after some time (case 5).

B. Analysis

We check that a UAV never returns to base under harsh weather (a safety property). To formalise this property, we represent state formulae by bigraph patterns: $\varphi_1 \stackrel{\text{def}}{=} \mathsf{B}(\text{“harsh\_weather”})$ and $\varphi_2 \stackrel{\text{def}}{=} \mathsf{B}(\text{“returned”})$. Using CTL, we have a property $\neg\mathbf{E}[\mathbf{F}(\varphi_1 \wedge \neg\varphi_2 \wedge (\mathbf{XX}\varphi_2))]$ that check there does not exist any agent behaviours in any environment that when the weather is harsh and UAV has not returned to base, the UAV is believed to be returned in two states[7] afterwards, *i.e.* we check that it does not travel through the bad weather.

However, this safety property does not hold for design in Fig. 5. To aid in addressing this problem, the violated states can be automatically located (by PRISM model checker). For debugging, the graphical output of each state in the transition system provided by BigraphER provides a diagrammatic representation of each state enabling us to locate subtle design issues. In this case, there are two failing situations. (1) when the weather becomes harsh, the agent can execute plans on line 5 or 6 to activate parking to avoid any potential damage. However, before the completion of parking (*e.g.* the execution of the action wait), the truth of detection which happens to hold can makes goal(detection, e_patrol_task, false) succeed, leading the agent to proceed returning prematurely without fully handling the negative environment. (2) occurs when the weather becomes harsh shortly after the UAV completes a detection, causing the UAV to return under harsh weather. To fix agent design flaws, we need to add ¬harsh_weather to the success condition of goal(detection, e_patrol_task, false) (blue in Fig. 6) for first case. For second case, we also require a declarative goal structure for the return task (red in Fig. 6). The property now holds for new agent design in Fig. 6.

We find that both designs in Fig. 5 and Fig. 6 can lead to a loop in the agent behaviours. Such a loop includes two situations: (1) persistent patrolling when no object is available to be detected and (2) whenever the agent is about to patrol or return, the weather becomes harsh so that the agent has to wait for the weather to be normal again. We can denote the bigraph pattern for the completion of the given intention as $\varphi_3 \stackrel{\text{def}}{=} \mathsf{Intent.1}$ if and only if it is the only intention in the base (which is our case), that checks that along all paths eventually the intention is completed either with success or with failure.

We also check that whenever a new event is requested from the environment, it will be responded by the agent eventually. We denote the bigraph pattern for the presence of new event request in an environment $\varphi_4 \stackrel{\text{def}}{=} \mathsf{Environment.(e\_patrol\_init \mid id)}$ and the successful assimilation of such event in external event set (a.k.a. desires) as $\varphi_5 \stackrel{\text{def}}{=} \mathsf{Desires.(e\_patrol\_init \mid id)}$ where the symbol id (called a site in bigraphs) stands for the part of model that is abstracted away. We can have the property $\mathbf{A}[\varphi_4 \implies \mathbf{F}\varphi_5]$ which checks that along all paths, if a new event is requested (*i.e.* φ_4 holds), this implies that eventually it will be added in the desires. To check that the agent actually committed to progressing this desire (leaving empty desire set in this case), we have the property $\mathbf{A}[\varphi_5 \implies \mathbf{F}\varphi_6]$ where $\varphi_6 \stackrel{\text{def}}{=} \mathsf{Desires.1}$. Both designs satisfy both of these properties.

A summary of these property checking is given in Table I. It also details the transition system that was used in the evaluation of each property: the number of states and

[7]When an action is executed by an agent (one $\mathbf{X}$), it requires another step to perceive the effects of action (another $\mathbf{X}$). Hence, two $\mathbf{X}$s are needed.

	Design in Fig. 5	Design in Fig. 6
Saftey Property	False	True
Completion Property	False	False
Response Property	True	True
Commitment Property	True	True
States	167	282
Transitions	242	373
Build time (s)	54.05	128.89
Rule applications	1306	2152

Table I: Properties checked: where safety property is $\neg\mathbf{E}[\mathbf{F}(\varphi_1 \wedge \neg\varphi_2 \wedge (\mathbf{X}\mathbf{X}\varphi_2))]$, completion property $\mathbf{A}[\mathbf{F}\varphi_3]$, response property $\mathbf{A}[\varphi_4 \implies \mathbf{F}\varphi_5]$, and commitment property $\mathbf{A}[\varphi_5 \implies \mathbf{F}\varphi_6]$.

transitions, build time (which are in the order of minutes), and rule applications. The rule applications are the number of applications of reaction rules, including instantaneous reaction rules—an advanced feature of BigraphER—that allows agents to progress an intention without showing all sub-steps. For example, it includes environment revision, where we see only final output of a step of executing an action by an agent.

VI. Related Work

When modelling environments for BDI agents, most existing work, similarly to this work, separates the specification of environments from the agent designs. For example, the JaCaMo platform [18] includes an artifact-based framework to allow programming and executing virtual environments. While agent computational simulations are essential, they can, by their nature, only analyse one possible run of agent behaviours in one environment.

Verifying BDI agent behaviours through model checking and theorem proving has also been explored. For example, the work [19] (resp. [20]) applys the Java PathFinder model-checker (resps. Isabelle/HOL proof assistant) to verify BDI programs. However, none of them has addressed how the environment should be transitioned from state to state and they only examine agent behaviours in a subset of environment. On the contrary, we support the analysis of all possible agent behaviours in all possible environments.

VII. Conclusions

A computational modelling and verification framework for BDI-agents can aid design-time specification by allowing us to reason about the behaviour of rational agents operating in dynamic environments, *e.g.* those that feature to changes of external world situations (harsh weather etc.).

We have provided a formalisation of an environment, including the sensor information, incoming external events which the agent needs to respond to, and the external dynamics (relative to an agent) in such an environment. The computational modelling of a dynamic environment is enabled by an extension to the CAN language (that formalises the behaviour of a classical BDI agent). The extended semantics are executable (via bigraphs) to allow both the development of agent designs and the specification of dynamic environments, exposing any potentially anomalous agent behaviour in any environment.

Through an UAV example, we have shown it is possible to reason about agent behaviours in all possible environments. In particular, we found that our approach can aid automatically identifying subtle agent design flaws rendered under some dynamic situations. The future work is to investigate uncertain environments to support numerical analysis, *e.g.* the probability of completing an intention in adversarial conditions.

Acknowledgements

This work is supported by the EPSRC, under PETRAS SRF grant MAGIC (EP/S035362/1) and S4: Science of Sensor Systems Software (EP/N007565/1).

References

[1] A. S. Rao, "AgentSpeak (L): BDI agents speak out in a logical computable language," in *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*. Springer, 1996, pp. 42–55.

[2] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, "Declarative and procedural goals in intelligent agent systems," in *Conference on Principles of Knowledge Representation and Reasoning*, 2002.

[3] K. V. Hindriks, F. S. D. Boer, W. V. d. Hoek, and J.-J. C. Meyer, "Agent programming in 3APL," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357–401, 1999.

[4] M. Winikoff, "JACK intelligent agents: an industrial strength platform," in *Multi-Agent Programming*, 2005, pp. 175–193.

[5] R. H. Bordini *et al.*, "Programming multi-agent systems in AgentSpeak using Jason," vol. 8. John Wiley & Sons, 2007.

[6] L. Royakkers and R. van Est, "A literature review on new robotics: automation from love to war," *International journal of social robotics*, vol. 7, no. 5, pp. 549–570, 2015.

[7] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge, "Verifying multi-agent programs by model checking," *Autonomous Agents and Multiagent Systems*, vol. 12, no. 2, pp. 239–256, 2006.

[8] R. Milner, *The space and motion of communicating agents*. Cambridge University Press, 2009.

[9] B. Archibald, M. Calder, M. Sevegnani, and M. Xu, "Modelling and verifying BDI agents with bigraphs," *Science of Computer Programming*, vol. 215, p. 102760, 2022.

[10] M. Kwiatkowska *et al.*, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.

[11] S. Sardina and L. Padgham, "Goals in the context of BDI plan failure and planning," in *International Conference on Autonomous Agents and Multiagent Systems*, 2007, pp. 16–23.

[12] M. Sevegnani, M. Kabác, M. Calder, and J. A. McCann, "Modelling and verification of large-scale sensor network infrastructures," in *Conference on Engineering of Complex Computer Systems,*, 2018, pp. 71–81.

[13] S. Benford, M. Calder, T. Rodden, and M. Sevegnani, "On lions, impala, and bigraphs: modelling interactions in physical/virtual spaces," *ACM Transactions on Computer-Human Interaction*, vol. 23, pp. 1–56, 2016.

[14] M. Bundgaard and V. Sassone, "Typed polyadic pi-calculus in bigraphs," in *International Conference on Principles and Practice of Declarative Programming*, 2006, pp. 1–12.

[15] M. Sevegnani and M. Calder, "BigraphER: Rewriting and analysis engine for bigraphs," in *the 28th International Conference on Computer Aided Verification*, 2016, pp. 494–501.

[16] S. Sardina and L. Padgham, "A BDI agent programming language with failure handling, declarative goals, and planning," in *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 1. Springer, 2011, pp. 18–70.

[17] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, 1981, pp. 52–71.

[18] O. Boissier, R. H. Bordini, J. Hubner, and A. Ricci, *Multi-agent oriented programming: programming multi-agent systems using JaCaMo*. MIT Press, 2020.

[19] L. A. Dennis, M. Fisher, M. P. Webster, and R. H. Bordini, "Model checking agent programming languages," *Automated software engineering*, vol. 19, no. 1, pp. 5–63, 2012.

[20] A. B. Jensen, "Machine-checked verification of cognitive agents," in *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 2022, pp. 245–256.

Zero-Shot Object Detection with Multi-label Context

Yongxian Wei, Yong Ma
School of Computer Science and Engineering, Nanjing University of Science and Technology
Nanjing, China
{wei_yx,mayong}@njust.edu.cn

ABSTRACT

Zero-shot detection (ZSD) , the problem of object detection when training and test objects are disjoint, i.e. no training examples of the target classes are available. ZSD increasingly gains importance for large scale applications because collecting and labeling sufficient data is extremely hard. In this paper, inspired from human cognitive experience, we propose a simple but effective Multi-label Context (MLC) framework to facilitate the detection ability for both seen and unseen objects by mining contextual cues. We design a multi-label classifier which leverages the holistic image-level context to learn object-level concepts. Then, novel RoI features are generated by exploiting context information beneath both whole images and interested regions. Moreover, background dynamic generator (BDG) can reduce the confusion between background and unseen classes. Our extensive experiments show that MLC outperforms the current state-of-the-art methods on MS-COCO.

Index Terms— Zero-Shot Object Detection, Multi-label Learning, Context Embedding, Computer Vision

1. INTRODUCTION

While object detection methods based on deep learning have achieved great progress over the last few years [1, 2, 3, 4, 5], these gains can be attributed to the availability of the fully supervised training data. Although researchers have struggled to acquire larger datasets with a broader set of categories, the processing procedure is time consuming and tedious. Furthermore, it is hard to collect enough training data for rare categories. Zero-shot learning (ZSL) has been proposed to address the problem for reasoning unseen classes [6, 7, 8], Traditional ZSL researches mainly focus on the classification of unseen objects and achieve high classification accuracy [7]. However, there is still a big gap between ZSL settings and real-world scenes. ZSL only focuses on identifying unseen objects, not detecting them. For example, most of datasets used as ZSL benchmark have only one dominant object per image [9, 10], while in real-world, various objects may appear in a single image without being precisely localized. To close this gap, [11] introduced a new "zero-shot object detection" (ZSD) problem setting method, which aims at detecting objects seen during training as well as detecting unseen classes as and when they appear at test-time.

(a) Baseline (b) Ours

Fig. 1. Motivation and example results of our MLC framework. By incorporating discriminative context information, the semantic features of "car" are strong evidences for detecting objects that are highly dependent on context information, such as "traffic light".

Existing ZSD approaches mainly focus on learning a visual-semantic correspondence based on intrinsic properties of the target objects by the means of human-defined attributes or distributed representations learned from text corpora. They only focus on local information near an object's region of interest while ignoring rich contextual information within the image, which has been shown to benefit the object detection performance [12, 13, 14].

We therefore propose a novel framework named Multi-label Context (MLC) for ZSD. In this paper, we revisit the RoI features in region-based detectors from the perspective of context information embedding. Our key motivation is that while each RoI in very deep CNNs may have a very large theoretical receptive field which usually spans the whole input image [1]. However, the effective receptive field [15] may only occupy a fraction of the entire theoretical receptive field, making the RoI features insufficient for characterizing objects that are highly dependent on context information. We use a simple but effective process to generate contextual RoI fea-

DOI reference number: 10.18293/SEKE2022-012

tures by exploiting embedded multi-label context information beneath both whole images and interested regions, which are also complementary to conventional RoI features.

MLC learns from the cognitive science about how humans reason objects through semantic information. Humans can learn the mapping relationship between vision objects and semantic description from seen objects and transfer it to detect unseen objects. In addition, conventional object detection approaches generally tend to relegate unseen objects into the background leading to missed detection of unseen objects. Previous works [16, 11] used the word-vector of the "background" word to represent background class. Due to the rough word-vector for background class used in detector head is inability to exactly represent the complex background, MLC develops a component denoted as background dynamic generator (BDG) to learn an appropriate word-vector for background class. Our study shows that replacing the rough background word-vector in detector head with the new one learned from BDG can effectively increase the recall rate of unseen classes.

In summary, the contributions of this paper are three-fold: (i) we develop a novel ZSD approach that adaptively exploits the whole image context to learn discriminative features for context-dependent object categories; (ii) to the best of our knowledge, it is the first time to introduce multi-label learning into ZSD task; (iii) extensive experiments on two different MS-COCO splits show significant performance improvement on the existing ZSD benchmarks.

2. METHOD

2.1. Problem Formulation

We begin by defining the problem and then present our approach. We denote the set of all classes as $C = C_S \cup C_U$, where C_S denotes the set of seen classes and C_U denotes the set of unseen classes, and $C_S \cap C_U = \phi$. Each image is denoted as $I \in \mathbb{R}^{w \times h \times 3}$, with corresponding bounding boxes and ground truth labels denoted as $b_i \in \mathbb{N}^4$ and $y_i \in C$ respectively. Let D_S denotes the training dataset, which only contains the objects belonging to C_S to train the network and use the unseen classes objects dataset D_U to evaluate the detection performance for unseen classes. For GZSD setting, the test dataset D_T contains objects from both seen and unseen classes ($c \in C = C_S \cup C_U$).

2.2. MLC Framework

The overall framework of our MLC consists of four components: Multi-Label Head for advancing the feature learning of the objects that are highly dependent on larger context clues, contextual RoI features generated by fusing both instance-level and global-level information derived from Multi-Label Head, BDG for generating suitable background word-vector, and Zero-Shot Head for classifying the extracted objects into seen and unseen classes and locating them. The details are indicated in Figure 2.

2.2.1. Multi-Label Head

In parallel with the RPN branch, we exploit Multi-Label Head upon the detection backbone, enabling the backbone to learn object-level concepts adaptively from global-level context. It is worth mentioning that Multi-Label Head does not require additional annotations, as the image-level labels can be conveniently obtained by collecting all instance-level categories in an image. Specifically, we first apply a 3×3 convolution layer on the output of ResNet conv5 to obtain the input feature map, and then follow the practice in [17] to employ both global max-pooling (GMP) and global average-pooling (GAP) for feature aggregation. Formally, let $\mathbf{X} \in \mathbb{R}^{d \times w \times h}$ denote the input feature map, where d is the channel dimensionality, w and h are the width and height, respectively. Then, the multi-label classifier is constructed by N_S binary classifiers for all categories:

$$\hat{\mathbf{y}} = f_{CLS}(f_{GMP}(\mathbf{X}) + f_{GAP}(\mathbf{X})) \in \mathbb{R}^{N_S}, \quad (1)$$

where N_S denotes the number of seen classes, each element of $\hat{\mathbf{y}}$ is a confidence score (logits), and f_{CLS} is binary classifier modeled as one fully-connected layer. We assume that the ground truth label of an image is $\mathbf{y} \in \mathbb{R}^{N_S}$, where $y_i = \{0, 1\}$ denotes whether object of category i appears in the image or not. The multi-label loss can be formulated as follows:

$$\mathcal{L}_{MLL} = -\sum_{i=1}^{N_S} y_i ln(\frac{1}{1+e^{-\hat{y_i}}}) + (1-y_i) ln(\frac{e^{-\hat{y_i}}}{1+e^{-\hat{y_i}}}), \quad (2)$$

2.2.2. Contextual RoI Feature Generation

With the purpose of leveraging larger context, We apply RoIAlign [4] with proposals generated by RPN on the context-embedded feature map X to obtain RoI features:

$$\mathbf{x}_{global} = f_{RoIAlign}(\mathbf{X}; w, h) \in \mathbb{R}^{d \times 7 \times 7}, \quad (3)$$

where $f_{RoIAlign}$ is the RoIAlign operation and w and h are the width and height of the input image, respectively. As the resulting RoI feature $\mathbf{x}_{global}$ absorbs rich context information from the context-embedded image feature $\mathbf{X}$, it is by nature complementary to the conventional RoI feature extracted from the feature pyramid network (FPN) [18]. To integrate our contextual RoI features $\mathbf{x}_{global}$ into the detection pipeline, it is natural to fuse them with the original RoI features extracted from the feature pyramid network (FPN) with element addition. Formally, let $\mathbf{x}_{instance}$ denote the original RoI feature extracted from FPN, and $\mathbf{x}_{fusion}$ denote the fused RoI feature, then we have:

$$\mathbf{x}_{fusion} = \mathbf{x}_{global} + \mathbf{x}_{instance} \in \mathbb{R}^{d \times 7 \times 7}, \quad (4)$$

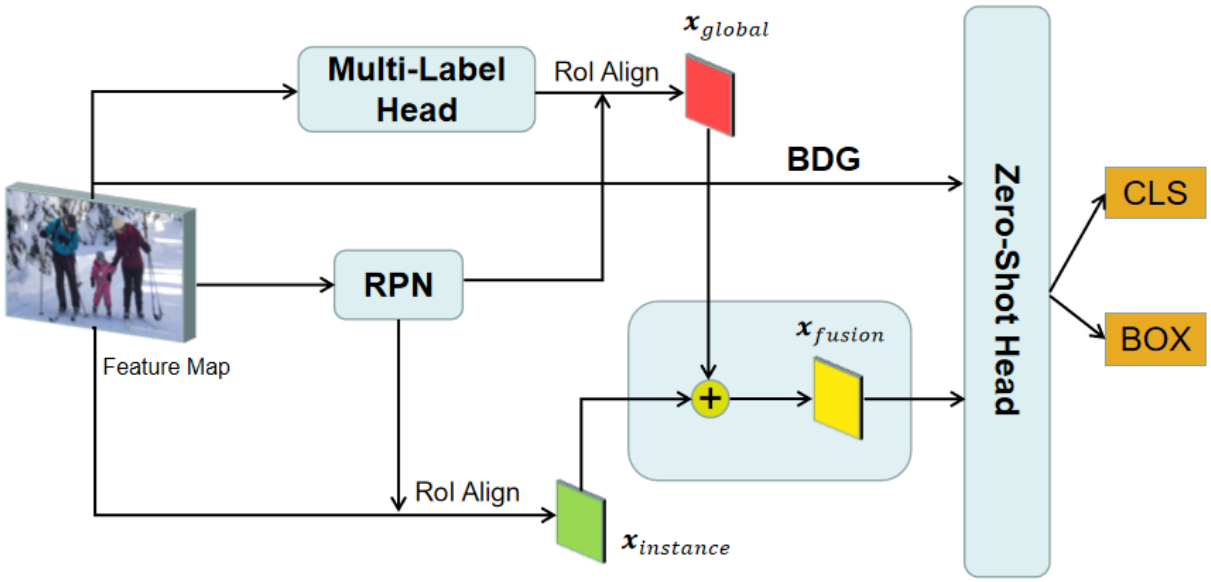


Fig. 2. The architecture for MLC. After acquiring feature map from the backbone, Multi-Label Head enables the network to learn object-level concepts from global-level context. Then, contextual RoI features which are complementary to conventional RoI features, are generated by fusing both instance-level and global-level information. Finally, the Zero-Shot Head uses the $\mathbf{x}_{fusion}$ and BDG to locate and classify the seen and unseen objects, respectively.

As shown in Figure 2, the fused feature map $\mathbf{x}_{fusion}$ is then fed into the Zero-Shot Head to produce refined bounding boxes and classification scores.

2.2.3. *Background Dynamic Generator*

We set a fully connected layer called $\mathbf{v}_b$ without bias and make it trainable. $\mathbf{v}_b$ is used to represent vector for background class, which is initialized with the mean word vectors for all seen classes. BDG will update $\mathbf{v}_b$ during training so that we can learn a new word-vector $\mathbf{v}_b$ for background class. During training, we feed the visual features derived from the backbone network to the BDG branch and get the background binary classification score. The calculation process is formulated as follows:

$$c = \frac{1}{1 + e^{-\mathbf{x}^T \mathbf{v}_b}}, \tag{5}$$

specifically, $T \in \mathbb{R}^{N \times d}$ is an FC layer which is used to adjust the dimension of input objective feature to fit d, i.e. the dimension of word vector.

2.2.4. *Zero-Shot Head*

The main idea for our Zero-Shot Head is learning the relationship between visual and semantic concepts from seen classes data and transferring it to detect unseen objects. To this end, we replace the classification branch in Faster R-CNN with a new semantic-classification branch. Keeping the non-trainable seen class word vectors W_S, we allow projection of the visual feature $\mathbf{x}_{fusion}$ to the word embedding space to calculate classification scores P_S. In inference, we follow [11] to use an additional procedure to calculate the classification scores for unseen classes. The process can be briefly demonstrated as follows:

$$P_U = (P_S W_S^T) W_U, P_S = T_e(\mathbf{x}_{fusion}) W_S, \tag{6}$$

where, W_U contains unseen class word vectors. So we can get the scores by performing the matrix multiplication of the semantic feature and W_U.

2.2.5. *Loss Function*

The whole loss function $\mathcal{L}_{MLC}$ for our end-to-end network has four components:

$$\begin{aligned} \mathcal{L}_{MLC} &= \mathcal{L}_{MLL} + \mathcal{L}_{RPN} + \mathcal{L}_{BDG} + \mathcal{L}_{ZSH}, \\ \mathcal{L}_{BDG} &= -(c \log(\hat{c}) + (1 - c) \log(1 - \hat{c})), \\ \mathcal{L}_{ZSH} &= -\sum_{i=1}^{N_S} P_{S,i} \log(P_{S,i}) + l_1(\mathbf{r}, \hat{\mathbf{r}}), \end{aligned} \tag{7}$$

where $\mathcal{L}_{ZSH}$ is the losses for Zero-Shot Head and it contains smooth l_1 regression loss. All loss terms are considered equally important, without extra hyper-parameters to characterize the trade-off between them, which reveals MLC is generalized and not trick.

Table 1. ZSD performance of Recall@100 and mAP with different IoU thresholds on MS COCO dataset.

Method	Seen/Unseen	Recall@100			mAP
		0.4	0.5	0.6	0.5
SB [16]	48/17	34.46	22.14	11.31	0.32
DSES [16]	48/17	40.23	27.19	13.63	0.54
TD [11]	48/17	45.50	34.30	18.10	-
PL [19]	48/17	-	43.59	-	10.10
Gtnet [20]	48/17	47.30	44.60	35.50	-
BLC [21]	48/17	51.33	48.87	45.03	10.60
Ours	48/17	**56.03**	**52.52**	**47.73**	**11.30**
PL [19]	65/15	-	37.72	-	12.40
BLC [21]	65/15	57.23	54.68	51.22	14.70
Ours	65/15	**60.11**	**57.81**	**52.49**	**15.70**

Table 2. Comparison of Recall@100 and mAP at IoU=0.5 under GZSD setting on MS COCO dataset. HM denotes the harmonic average for seen and unseen classes.

Method	Seen/Unseen	seen		unseen		HM	
		mAP	Recall	mAP	Recall	mAP	Recall
DSES [16]	48/17	-	15.02	-	15.32	-	15.17
PL [19]	48/17	35.92	38.24	4.12	26.32	7.39	31.18
BLC [21]	48/17	42.10	57.56	4.50	46.36	8.20	51.37
Ours	48/17	**47.26**	**71.46**	**5.39**	**50.92**	**9.68**	**59.46**
PL [19]	65/15	34.07	36.38	12.40	37.16	18.18	36.76
BLC [21]	65/15	36.00	56.39	13.10	51.65	19.20	53.92
Ours	65/15	**40.95**	**67.83**	**14.86**	**59.64**	**21.81**	**63.47**

3. EXPERIMENT

3.1. Dataset and Setting

We validate our proposed method on the widely used object detection dataset MSCOCO. This dataset is more challenging than Pascal VOC as it has 80 object classes, more small objects, and more complex background. Following the dataset splits of MSCOCO proposed in [16] and [19], we use both two splits of the dataset in experiments: (1) 48 seen classes and 17 unseen classes; (2) 65 seen classes and 15 unseen classes. Note that the seen classes and unseen classes are disjoint.

We use mAP and Recall@100 as the evaluation metrics, in which 100 means that only the top 100 detections are valid for evaluation. The experimental results are reported under ZSD (zero shot detection) and GZSD (generalized zero shot detection) benchmarks. The ZSD setting only requires the detection results of unseen objects, while for GZSD setting, it requires the model predict both the seen and unseen objects. GZSD is more challenging than ZSD, and more suitable for practical application.

Table 3. Ablation study of our method in different splits. ZSH means Zero-Shot Head and MLH means Multi-Label Head.

Seen/Unseen	ZSH	MLH	BDG	Recall/mAP		
				seen	unseen	HM
48/17	✓			65.1/40.7	43.1/4.5	51.8/7.7
	✓	✓		70.9/47.1	45.3/5.5	55.3/9.8
	✓	✓	✓	71.4/47.2	50.9/5.3	59.4/9.6
65/15	✓			60.6/32.0	54.3/12.7	57.3/18.2
	✓	✓		65.7/39.2	55.0/14.6	59.9/21.3
	✓	✓	✓	67.8/40.9	59.6/14.8	63.4/21.8

3.2. Comparison with Other Methods

We compare the performance for MLC with the state-of-the-art zero-shot detection approaches on both 48/17 and 65/15 splits of MSCOCO under ZSD and GZSD settings. For ZSD setting, we show the results in Table 1. Our method outperforms all other work and improves up to 30.38% and 20.09% of the Recall@100 metric over the 48/17 and 65/15 splits, respectively. Moreover, the improvement in mAP also shows that the contextual RoI feature has an effective discrimination ability to unseen class. For GZSD setting, we report the results in Table 2. MLC surpasses all previous works in terms of mAP and Recall@100 on both seen and unseen classes. The "HM" performance gain reveals that our method maintains a good balance between seen and unseen classes.

3.3. Ablation Study

We conduct a controlled study of our proposed method on GZSD evaluation. As shown in Table 3, the baseline method with Zero-Shot Head gives a foundation and achieves comparable mAP and recall at IoU = 0.5. Our method is able to consistently bring improvement on both seen and unseen categories. From these results, we can learn the significant effectiveness of the Multi-Label Head. We can also observe that BDG brings an improvement of 4.6% in terms of Recall@100 for unseen classes.

4. CONCLUSION AND FUTURE WORK

In this paper, we find that contextual information is quite important in zero-shot detection, hence we propose a novel framework to embed global-level context to advance the learning of context-dependent categories with the help of multi-label learning. In the experiment part, we described the extensive experiments which were conducted to demonstrate

the superiority of the proposed model, and investigated the effectiveness of different components. In the future, we would like to find a better approach to obtain the semantic feature since traditional word vectors like word2vec are noisy.

5. REFERENCES

[1] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[5] Joseph Redmon and Ali Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[6] Abhijit Bendale and Terrance E Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[7] Ziming Zhang and Venkatesh Saligrama, "Zero-shot learning via joint latent similarity embedding," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 6034–6042.

[8] Yong Ma, Huaqi Mao, Haofeng Zhang, and Wenbo Wang, "Prototype relaxation with robust principal component analysis for zero shot learning," *IEEE Access*, vol. 8, pp. 170140–170152, 2020.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[10] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona, "Caltech-ucsd birds 200," 2010.

[11] Shafin Rahman, Salman Khan, and Fatih Porikli, "Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 547–563.

[12] Xiaowei Hu, Lei Zhu, Chi-Wing Fu, Jing Qin, and Pheng-Ann Heng, "Direction-aware spatial context features for shadow detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7454–7462.

[13] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta, "Iterative visual reasoning beyond convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7239–7248.

[14] Haowen Deng, Tolga Birdal, and Slobodan Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 195–205.

[15] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4905–4913.

[16] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran, "Zero-shot object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 384–400.

[17] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[19] Shafin Rahman, Salman Khan, and Nick Barnes, "Improved visual-semantic alignment for zero-shot object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 11932–11939.

[20] Shizhen Zhao, Changxin Gao, Yuanjie Shao, Lerenhan Li, Changqian Yu, Zhong Ji, and Nong Sang, "Gtnet: Generative transfer network for zero-shot object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 12967–12974.

[21] Ye Zheng, Ruoran Huang, Chuanqi Han, Xi Huang, and Li Cui, "Background learnable cascade for zero-shot object detection," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

Modified Communication Parallel Compact Firefly Algorithm and Its Application

1st Jianpo Li
School of Computer Science
Northeast Electric Power University
Jilin, China
jianpoli@163.com

2nd Geng-Chen Li
School of Computer Science
Northeast Electric Power University
Jilin, China
605544372@qq.com

3rd Jeng-Shyang Pan*
College of Computer Science and Engineering
Shandong University of Science and Technology
Qingdao, China
jspan@cc.kuas.edu.tw

4th Min Gao
School of Computer Science
Northeast Electric Power University
Jilin, China
15822553806@163.com

***Abstract*—Coverage is an important indicator to measure the monitoring quality of Wireless Sensor Network (WSN). As a NP-hard problem, it is a mainstream method to introduce swarm intelligence algorithm to solve it. After analyzing the traditional Firefly Algorithm (FA), aiming at the defects of this algorithm, this paper proposes Modified Communication Parallel Firefly Algorithm (MCPFA) family algorithm, which improves the performance of the algorithm to a certain extent. On this basis, the compact optimization method is introduced and the Modified Communication Parallel Compact Firefly Algorithm (MCPCFA) family algorithm is proposed to further improve the overall function of traditional FA. The proposed algorithm is tested by several classical functions in CEC2013 test function set to verify the performance of the algorithm. Finally, a WSN node deployment scheme based on MCPCFA family algorithm is proposed to improve the network coverage. Through simulation experiments, compared with the traditional Partial Swarm Optimization (PSO), FA, Parallel FA (PFA) and Compact FA (CFA), MCPCFA family algorithm shows the best performance in WSN network layout.**

***Index Terms*—firefly algorithm, modified parallel strategy, compact optimization method, WSN, coverage optimization**

I. Introduction

Swarm intelligence algorithm is an optimization method proposed by researchers based on the research of biological group behavior and physical phenomena in nature. In recent decades, many scholars at home and abroad have proposed a variety of swarm intelligence algorithm. These algorithms are inspired by biological evolution or natural biological habits. Therefore, each algorithm has its own rules and characteristics. Yang Xin-she simulated the living habits of fireflies and proposed FA in [1]. As a swarm intelligence algorithm, it can also be free from the nature of optimization problems. These algorithms or their variants are used to solve optimization problems in various fields. Like other swarm intelligence algorithms, FA and its variants are used to solve optimization problems in a variety of domains.

DOI reference number: 10.18293/SEKE2022-019

Internet of things and big data are hot research and application fields in the intelligent era. As the core infrastructure of Internet of things and big data, WSN have been widely studied and deeply developed in recent years [2]. At present, WSN have been integrated into life and widely used in various fields [3]. Node deployment is one of the basic problems that must be properly solved for any type of WSN application to achieve the expected quality of service. Therefore, the research on the layout of WSN is particularly important. The traditional 2D coverage research is not enough to meet the actual needs, so the coverage research in 3D environment has attracted extensive attention of researchers. As a NP-hard problem, the introduction of evolutionary algorithm solves some problems to a certain extent.

The 3-D curved surface deployment problem is a special situation in the three-dimensional space deployment. It is closer to the application scenarios in the real world, such as volcano monitoring, building structure monitoring, and disaster rescue monitoring. Sensors can only be placed on the surface of the three-dimensional terrain, and cannot be arbitrarily deployed in the air or in the mud. The research on the area coverage of 3-D curved surface is to monitor the area of the mouth mark on the terrain surface such as the mountain surface and the building surface. Due to the complex shape of the terrain surface and the changeable environment, it is very difficult to monitor the events or important parameter information of the three-dimensional surface. Therefore, the research on the deployment of 3D surface nodes has very important practical application value.

In this paper, our goal is to analyze the shortcomings of FA and make corresponding improvements, so as to improve the performance of the algorithm. Because the traditional FA has the problem of poor convergence performance and the algorithm is easy to fall into the local optimum, this paper proposes three modified communication parallel strategy. Then, FA has the problem of high time complexity. For this problem, this paper proposes a compact scheme combined

with the FA to reduce the time complexity of the algorithm. And combined with the parallel FA family algorithms, the proposed compact scheme is introduced into each group, and the MCPCFA family algorithms are proposed, which further improves the overall performance of FA. Using the improved version of FA proposed in this paper, MCPCFA family algorithm optimizes the node deployment strategy of WSN and finds the node layout mode with the maximum coverage. The main contributions of this paper are summarized as follows.

- MCPFA family algorithm is proposed, and the population number of MCPFA is updated to further improve the performance of FA.
- The performance of the proposed MCPFA is analyzed. In order to use less memory to simulate the operation of MCPFA, the compact idea is added, and the MCPCFA is proposed to improve the overall function of FA.
- In this paper, CEC2013 test function set is used to test and analyze the performance of the proposed MCPCFA family algorithm, which verifies our theory. The data results show that compared with other comparison algorithms, MCPCFA shows strong ability.
- This paper introduces the traditional WSN coverage model, 0-1 model. At the same time, MCPCFA family algorithm was used to optimize the model. Through the analysis of simulation results, it is confirmed that the optimization ability of the proposed method is better than other methods. The superiority and applicability of MCPCFA in this field are also verified.

The rest of the arrangements are as follows. The second part reviews the FA algorithm and proposes the MCPCFA algorithm. The third part analyzes the experimental performance of the proposed MCPCFA algorithm. Section IV introduces the 0-1 coverage model of WSN. Section V discusses the application of MCPCFA in WSN coverage in detail. Finally, the work of this paper is summarized.

II. Modified Communication Parallel Compact Firefly Algorithm

A. Firefly Algorithm

FA simulated the biological behavior of fireflies and assumed that each firefly was always glowing. In order to simulate the luminous behavior of fireflies, assuming that all fireflies no gender distinction, each a firefly is likely to be attracted to other individuals. In addition, the bright fireflies will attract darker fireflies, and their brightness is directly proportional to the fitness function value with its location. The specific steps of FA are as follows.

First of all, each a firefly in the fitness function value is calculated, and according to the results of this decision direction of movement of each individual.

Then, their individual attractiveness was calculated by (1).

$$\beta = \beta_0 \times e^{-\gamma r_{ij}^2} \tag{1}$$

Finally, each firefly moves its position according to the rules of (2).

$$x_{id}(t+1) = x_{id}(t) + \beta\left(x_{jd}(t) - x_{id}(t)\right) + \alpha \tag{2}$$

Where, I_0 is the maximum brightness of firefly individual. γ is the light intensity absorption coefficient. r_{ij} is the spatial distance between firefly i and j. β_0 is the maximum attraction. α is the step factor, which is a constant on $[0, 1]$.

B. Modified Communication Parallel Firefly Algorithm

Although the traditional FA has stronger global search ability than other algorithms, like most swarm intelligence algorithm, FA still has the disadvantage that it is easy to fall into the local optimal solution, resulting in the premature convergence of the algorithm in the running process. Therefore, aiming at this problem, based on the traditional parallel strategy [4], this paper proposes MCPFA family algorithm. Next, the new ideas proposed in this paper will be explained in detail.

Firstly, it is assumed that there are pop firefly individuals in the population, and the maximum number of iterations is $iter_{\max}$. Before the algorithm starts to run, the whole firefly population is clustered, and the population is divided into g groups. Then there will be pop/g particles in each group after clustering. In addition, it is also necessary to set up an information exchange mechanism to exchange information every time the number of iterations t of the algorithm reaches an integer multiple of R, and the exchange strategy will be divided into two parts according to the different search stages of the algorithm. After this operation, the population will return to the optimization state at the position after information exchange to continue the optimization work. The communication strategies in the two different search stages are explained below. It is worth mentioning that two different search stages will be distinguished when $t = \frac{iter_{\max}}{2}$, the first stage before and the second stage after. That is, when $t < \frac{iter_{\max}}{2}$, the algorithm is considered to be in the first stage, while when $t \geq \frac{iter_{\max}}{2}$, the algorithm is considered to be in the second stage.

a) Communication strategy in the first stage: Like most swarm intelligence algorithm, FA also has some disadvantages, such as poor convergence and easy to fall into local optimality. Therefore, in this part of the work, we will improve FA to improve the convergence and accuracy of the algorithm. In addition to the different communication strategies used, MCPFA family algorithm will use the same grouping method and preparations before communication. The specific operations are as follows.

- Modified Communication Parallel Firefly Algorithm-Best (MCPFA-B) After the grouping operation, all particles have entered the parallel working state. Every time the number of iterations t reaches R, the action of information exchange will occur. Before that, the particles in each group will be sorted according to the size of fitness function value, and the arranged population will be divided into A and B parts. Among them, A represents the

particles whose fitness function value is in the first half. Then, the particle order after the A part is $x_{Abest}\,(x_{A1}) > x_{A2} > x_{A3} > \ldots > x_{Aworst}\left(x_{A\frac{pop}{2}}\right)$. The B part represents the particles whose fitness function value is in the last half. Then, the particle order after the B part is $x_{Bbest}\,(x_{B1}) > x_{B2} > x_{B3} > \ldots > x_{Bworst}\left(x_{B\frac{pop}{2}}\right)$. Among them, it is worth noting that the fitness function value of x_{Aworst} is better than that of x_{B1}. The reason for this is that the algorithm itself has the disadvantage of poor convergence. Using the best particle to replace the worst particle can speed up the convergence speed of the algorithm to a certain extent and make the algorithm find the global optimal solution faster.

$$\begin{aligned} x_{Abest}\,(x_{A1}) &\rightarrow x_{Aworst}\left(x_{A\frac{pop}{2}}\right) \\ x_{Bbest}\,(x_{B1}) &\rightarrow x_{Bworst}\left(x_{B\frac{pop}{2}}\right) \end{aligned} \tag{3}$$

Where x_{A1} and x_{B1} represent the position of the particle with the best fitness function value in A and B respectively. x_{Aworst} and x_{Bworst} represent the position of the particle with the worst fitness function in A and B respectively. $\rightarrow$ expresses the meaning of substitution. After information exchange, the replaced particles will be optimized again in the updated position. When the number of iterations t reaches an integral multiple of R again, information exchange will be carried out again until the end of the first search phase. Fig. 1 shows a schematic diagram of this optimization scheme.

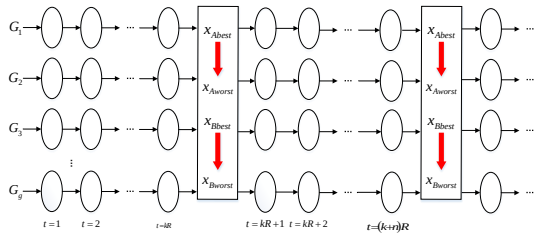


Fig. 1. Communication strategy of the first stage of MCPFA-B.

- Modified Communication Parallel Firefly Algorithm-Average (MCPFA-A) Like MCPFA-B, the algorithm performs grouping operation first, and then all particles enter the optimization state. When the number of iterations $t = kR$, the particles are sorted according to the fitness function value. However, different from the previous method, MCPFA-A will use the average value of the current population for information exchange in the operation of information exchange. As shown in (4), the average value of the position of the particles in A and B is calculated and recorded as x_{Aave} and x_{Bave}.

$$\begin{aligned} x_{Aave} &= \frac{x_{A1}+x_{A2}+\ldots+x_{Aworst}}{\frac{pop}{2g}} \\ x_{Bave} &= \frac{x_{B1}+x_{B2}+\ldots+x_{Bworst}}{\frac{pop}{2g}} \end{aligned} \tag{4}$$

After getting the average, the algorithm will replace the worst value in the part A with the average value in the part A. Part B is the same as part A. The specific operation is shown in (5).

$$\begin{aligned} x_{Aave} &\rightarrow x_{Aworst} \\ x_{Bave} &\rightarrow x_{Bworst} \end{aligned} \tag{5}$$

Where x_{Aave} and x_{Bave} represent the average position of all particles in part A and B respectively. Fig. 2 shows the schematic diagram of MCPFA-A optimization scheme in the first stage.

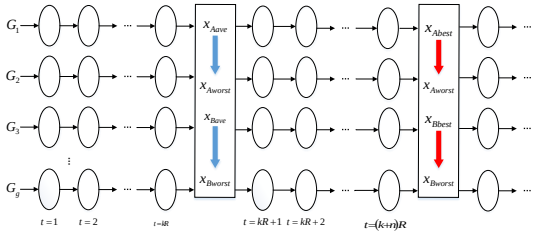


Fig. 2. Communication strategy of the first stage of MCPFA-A.

- Modified Communication Parallel Firefly Algorithm-Rand (MCPFA-R) Similar to the methods of MCPFA-B and MCPFA-A, MCPFA-R also needs to sort the fitness function values of fireflies in each group before information exchange, and divide the particles in each group into two parts: Part A and B. Then, MCPFA-R randomly selects one particle x_{Arand} and x_{Brand} in A and B respectively. These two particles are used to replace the worst fitness particles x_{Aworst} and x_{Bworst}. The specific operation is shown in (6).

$$\begin{aligned} x_{Arand} &\rightarrow x_{Aworst} \\ x_{Brand} &\rightarrow x_{Bworst} \end{aligned} \tag{6}$$

Where x_{Arand} and x_{Brand} represent the positions of randomly selected particles in parts A and B respectively. For ease of understanding, Fig. 3 shows the schematic diagram of MCPFA-R optimization scheme in the first stage.

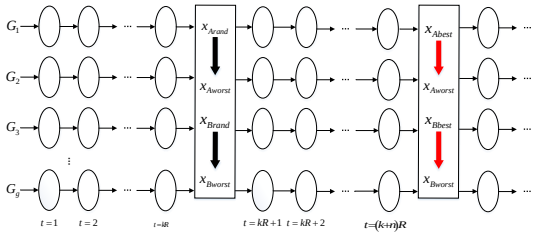


Fig. 3. Communication strategy of the first stage of MCPFA-R.

b) Communication strategy in the second stage: In the second stage, FA begins to enter the convergence stage, that is, the whole population begins to approach the global optimal solution. At this stage, the operation of information exchange still needs to be carried out. However, since the possibility of a better position in the solution space is relatively small, and the algorithm is in the convergence stage, the information exchange operation in the second stage will only use the optimal value in each group. At the same time, in order to improve the convergence speed of the algorithm, when the number of iterations reaches an integral multiple of M, the previously divided g group population is fused. The so-called fusion processing means that after each fusion, The number of groups of the population will be reduced to half of the previous one. At the same time, the latter half of the particles with relatively poor fitness function value in the fused group should be discarded and only the first half of the particles should be retained. In this way, the number of firefly individuals in the whole population will continue to decrease until there is only one group left, and the fusion operation will be stopped. The exchange of information will not be stopped until the end of the algorithm. The schematic diagram of the proposed idea in this stage is shown in Fig. 4.

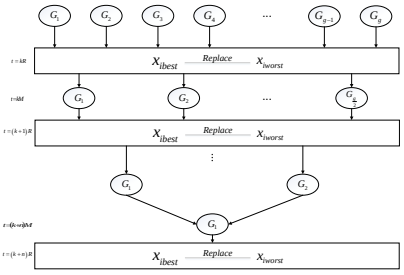


Fig. 4. Communication strategy of the second stage of MCPFA.

To sum up, MCPFA family algorithm proposed in this paper adds a modified parallel strategy on the basis of traditional FA, divides the population into g groups and optimizes at the same time. When the algorithm is in the first stage, whenever the number of iterations t reaches an integral multiple of R, three communication strategies are used to replace the position of the worst fitness function value particle x_{worst} in the group, so as to improve the global search ability of the algorithm. When the algorithm is in the second stage, in order to improve the convergence of the algorithm, the particle x_{best} with the best fitness function value in the group is used to replace x_{worst}. At the same time, whenever t reaches an integer multiple of M, the packets are fused. For example, the first group and the second group are fused, the third group and the fourth group are fused, and the latter half of the particles with relatively poor fitness function value are discarded to improve the computer memory occupied by the algorithm and improve the efficiency of the algorithm. In practical application, R and M will be set in advance.

C. Compact optimization method

This paper proposes MCPFA family algorithm, which improves the performance of the algorithm to a certain extent, but MCPFA family algorithm still has room to improve. For example, the traditional FA occupies too much computer memory space, and MCPFA is also troubled. In order to solve this problem, compact optimization idea is introduced in this study. This optimization method simulates the population behavior of the algorithm by establishing a population distribution probability. The most essential feature of compact method is that it has no actual population, but uses virtual population instead. The virtual population is a probability model, and they are encoded in a data structure, represented by the disturbance vector PV [5], [6].

$$PV = [\mu, \delta] \tag{7}$$

Where, μ and δ are two parameters, which are expressed as the mean and standard deviation of the PV vector respectively, t indicates the number of iterations of the current program μ and δ will vary in the Gaussian probability density function PDF and are limited to the region of $[-1, 1]$. At the same time, PDF is normalized so that its area is 1, and it is a uniform distribution of complete shape [7].

$$PDF_{\mu_i,\sigma_i}(x) = \frac{e^{-\frac{-(x-\mu_i)^2}{2\sigma_i^2}}\sqrt{\frac{2}{\pi}}}{\sigma_i\left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)} \tag{8}$$

$$\mu_i(t+1) = \mu_i(t) + \frac{1}{N_p}(winner_i - loser_i) \tag{9}$$

$$\sigma_i^2(t+1) = (\sigma_i(t))^2 + (\mu_i(t))^2 - (\mu_i(t+1))^2 + \frac{1}{N_p}\left(winner_i^2 - loser_i^2\right) \tag{10}$$

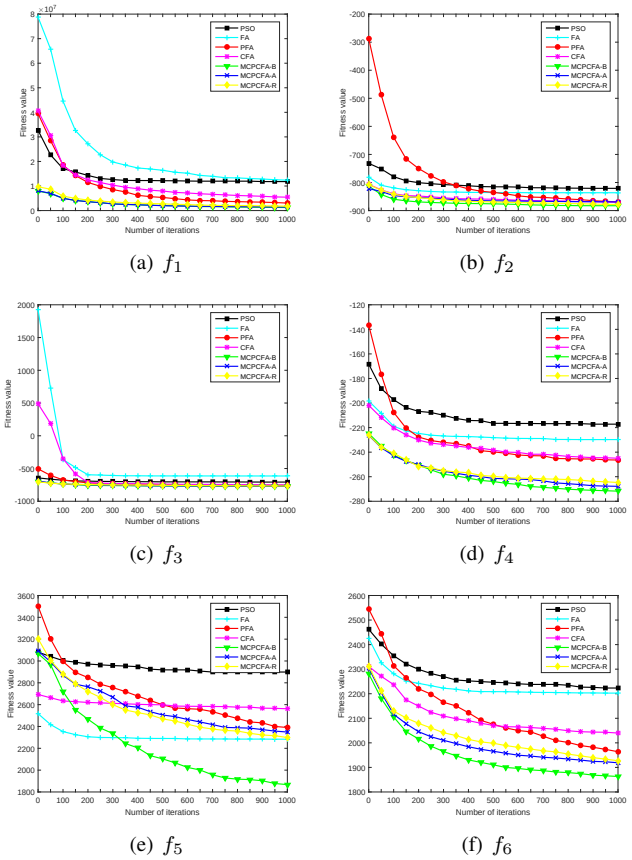
Finally, the time complexity of MCPCFA is theoretically analyzed to better introduce the MCPCFA algorithm proposed in this paper. the time complexity of the compression strategy FA is O($g\times d$), the time complexity of updating the optimal value is O(1), and the time complexity of the FA algorithm using the parallel strategy is O($g\times g$). So the above calculation complexity is max(O($g\times d$), O(1), O($g\times g$))=O($g\times d$), std. $d>g$. The computational complexity of the entire algorithm is O($time_{max}\times g\times d$), that is, the time complexity of MCPCFA is O($time_{max}\times g\times d$).

III. Test Experiments to Verify the Performance of the Proposed Algorithm

In order to verify the advantages of MCPCFA family algorithm proposed in this paper, several classical test functions from CEC2013 test function set will be used to test the performance of the algorithm in MATLAB2015b. At the same time, comparative experiments were carried out with PSO, FA, PFA and CFA. In order to ensure the fairness of the experimental results, the parameter settings of each algorithm will be set uniformly. Each test function will be run 30 times and the results will be averaged and compared. The population

number of all algorithms is set to 80 and the maximum number of iterations itermax is set to 1000. In PFA and MCPCFA family algorithm, the population was divided into 8 groups. In addition, make all teams exchange information every 20 iterations.

CEC2013 test function set includes Unimodal Functions, Basic Multimodal Functions and Composition Functions. We select two classical test functions from each category to test the performance of the algorithm. The results of this test are shown in Tab. I and Fig. 5.



(a) f_1 (b) f_2

(c) f_3 (d) f_4

(e) f_5 (f) f_6

Fig. 5. The results of test experiment

Through the above test experiments, we can clearly see that the MCPCFA family algorithm proposed in this study show better performance in these three types of problems. This also verifies the superiority of the improvement scheme proposed in this study.

To better illustrate the advantages of memory usage after introducing the parallel compression optimization method, we analyze the memory situation of MCPCFA and FA. A comparison of memory costs is shown in the Tab. II. The number of variables for the FA and MCPCFA algorithms can be obtained from the above update equations. In the Tab. II, the algorithm update equation consists of Eq. (1-2) and Eq. (7-10). Assuming that the number of particles is n, and n is a positive integer. The population size of MCPCFA is n. So the time complexity is 6×n×T×Iterations. Similarly, we can get the time complexity of FA as 2×N×T×Iterations. Although the update equation of MCPCFA is larger than that of FA, it has only n individuals and n is much smaller than N. So it consumes less memory than the original algorithm.

IV. 3D 0-1 Coverage Model of Wireless Sensor Network

The node deployment problem of WSN is necessary [9]. The traditional coverage problem is usually studied based on 2D plane, but if the sensor nodes are placed on 2D plane, the simulation experiment will be obviously different from the actual situation. Therefore, in this study, the sensor nodes are placed on the 3D terrain, and the terrain elements such as high and low-lying are added to the three-dimensional terrain, so as to more truly simulate the actual coverage problem. To sum up, the 3D 0-1 model and the terrain model proposed in this study will be described in detail below.

In general, the sensing model of sensor nodes is usually simplified to 0-1 model, that is, when a point in the area is covered by sensor nodes, it is recorded as 1. If it is not overwritten, it is recorded as 0. The most commonly used 0-1 sensing model is the sensing disk model. All points within the radius of a disk with a fixed length r centered on a sensor node are considered to be covered by the node. Assuming that the coordinate of a node i in the detected area is (x_i, y_i, z_i), the communication radius is r and the coordinate of the target pixel j is (x_j, y_j, z_j), the distance between the node i and the target pixel j is as follows.

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{11}$$

Use $O_{i,j}$ to represent the perception quality of node i to pixel j. When the position of the pixel j to be concerned is within the circle of the sensing range r of node i, it is considered that the perception quality of node i to pixel j is 1, that is, the perception of node i to pixel j is 1. Otherwise, when pixel j is outside the sensing range of node i, the perception of node i to pixel j is 0, Therefore, the mathematical expression is as follows.

$$O_{i,j} = \begin{cases} 1, d_{i,j} \leq r \\ 0, otherwise \end{cases} \tag{12}$$

The traditional 2D coverage model ignores the factors in the actual terrain. Although the problem is simplified, it can not be applied in practice. Therefore, during the experiment, the terrain elements of high and low-lying are added to the terrain, so that the model can more truly simulate the actual scene. The next simulation experiment will also use the model.

V. Application of MCPCFA family algorithm in WSN 3D coverage optimization

To solve the coverage problem is essentially to find the optimal deployment strategy. Different strategies have a significant impact on coverage, especially on 3D terrain. By optimizing the parameters of FA, using modified parallel strategy and introducing compact idea, this paper proposes MCPCFA family algorithm to improve the performance of FA. MCPCFA family algorithm performs well in solving basic multimodal functions, and the 0-1 coverage of WSN belongs to

TABLE I
COVERAGE OF DIFFERENT NUMBER OF NODES

Function	f_1	f_2	f_3	f_4	f_5	f_6
PSO	11806088	-820.806	-706.818	-217.204	2898.773	2223.218
FA	12371935	-836.479	-614.059	-229.762	2283.462	2202.544
PFA	3051820	-869.359	-755.468	-246.415	2390.835	1963.838
CFA	5453586	-870.054	-750.4	-244.972	2562.319	2039.94
MCPCFA-B	1234468	-882.79	-773.23	-271.77	1867.24	1863.02
MCPCFA-A	1434183	-869.116	-768.149	-267.83	2349.067	1918.792
MCPCFA-R	1778181	-878.305	-766.806	-264.703	2298.769	1927.079

TABLE II
THE TIME COMPLEXITY OF THE TWO ALGORITHMS

Algorithm	Particle	Memory Size	Computing Complexity	Use Equations
MCPCFA	n	6×n	6×n× T× Iteration	(1)(2)(7)(8)(9)(10)
FA	N	2× N	2× T× N× Iteration	(1)(2)(3)

this kind of problem. Therefore, the 0-1 coverage problem of WSN is effectively solved by the MCPCFA family algorithm.

The sensor node is set on the ground, so you only need to know the two coordinate values of a point to calculate the coordinates of the point. Therefore, the algorithm can optimize the deployment strategy by optimizing the location of any 2D sensor nodes. Each particle of the algorithm represents a deployment strategy.

Each individual updates their own location and calculates the fitness function value according to (13).

$$R(t) = \frac{1}{Z}\sum_{j=1}^{Z}\left(\sum_{i=1}^{P} O_{i,j}\right) \tag{13}$$

Where, $R(t)$ is the coverage at iteration t, p represents the number of sensor nodes, and Z represents the number of pixels in the simulation model made in this study. The purpose of this experiment is to find a node deployment strategy with the largest network coverage under the condition of existing hardware facilities.

The proposed algorithm is applied to the WSN 3D coverage model by using MATLAB 2015b simulation tool to verify the applicability of the proposed MCPCFA family algorithm in this field. Firstly, the sensor nodes are randomly distributed in this 3D terrain. In order to fully prove the performance of the proposed algorithm in 3D coverage, PSO, FA, PFA and CFA are used for comparative experiments. The number of sensor nodes is set to 30-55. The communication radius r of each node is set to 5-10m. The size of the monitoring area is 100m*100m. Sensor nodes are randomly distributed in the 3D terrain. The simulated 3-D terrain is shown in Fig. 6. The maximum number of iterations $iter_{\max}$ of the algorithm is set to 30. The initial population of pop is 80. Information exchange is conducted every three iterations of the program. At the same time, $t \leq 15$ is the first stage of the algorithm, and $t > 15$ enters the second stage of the algorithm. In other words, from $t = 18$, the firefly population will be fused between groups. Make the program fuse once every three iterations.

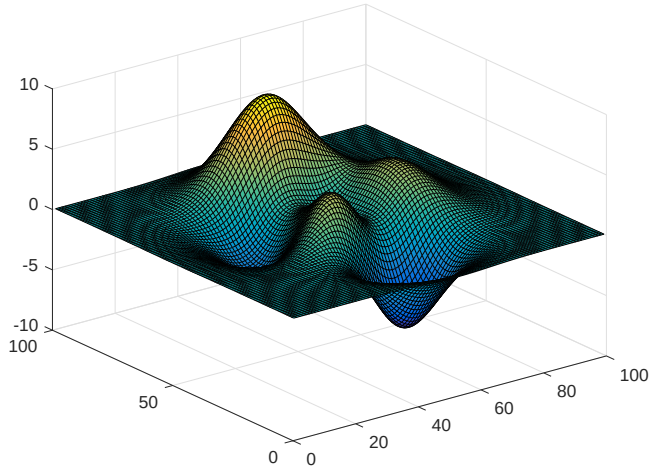


Fig. 6. Random deployment of sensor nodes on 3-D topographic maps

It is worth noting that MCPCFA family algorithm starts the fusion operation between groups when the number of iterations is $t = 18$, while $g = 8$ and $M = 3$ are set in this experiment, that is, the fusion action between groups occurs when $t = 18$, 21 and 24 respectively. Since then, the number of groups has been gradually fused from 8 groups to 1 group, there is no need for fusion operation. Next, the results of the two groups of simulation experiments will be summarized and analyzed.

A. Simulation experiment when the total number of sensor nodes is different

In this part, we will experiment with different number of sensor nodes. For fairness, we guarantee that other parameters remain unchanged. The total number of sensor nodes is set to

30, 35, 40, 45, 50 and 55 respectively, and the communication radius is uniformly set to 5m. The experimental results optimized by different algorithms are recorded in Tab. III.

TABLE III
COVERAGE OF DIFFERENT NUMBER OF NODES

Nodes	30	35	40	45	50	55
PSO	0.4611	0.5082	0.5541	0.5959	0.6359	0.6657
FA	0.4637	0.5176	0.5608	0.598	0.6389	0.6707
PFA	0.4688	0.5203	0.5642	0.6046	0.6434	0.6752
CFA	0.4697	0.5233	0.5647	0.6095	0.6462	0.6763
MCPCFA-B	0.4952	0.5432	0.5939	0.6354	0.6784	0.7612
MCPCFA-A	0.4891	0.5341	0.5845	0.6233	0.6658	0.6937
MCPCFA-R	0.4897	0.5408	0.5881	0.6289	0.6663	0.7008

It can be seen from the results in Tab. III that with the increase of the number of sensor nodes, the coverage of WSN is also improving. At the same time, when the number of nodes is 50, compared with traditional PSO, FA, PFA and CFA, the coverage optimized by MCPCFA-B is increased by 4.52%, 3.95%, 3.5% and 3.22% respectively. When the number of nodes is other values, the coverage has also been significantly improved. It can be seen that no matter how many sensor nodes are, mcpcfa finds the node deployment mode with the highest coverage through optimization.

B. Simulation experiment with different communication radius

In this part, we will experiment with different communication radius of sensor nodes. Similarly, for fairness, the total number of sensor nodes is uniformly set to 30, and other parameters are consistent. We set the communication radius as 5m, 6m, 7m, 8m, 9m and 10m respectively. The experimental results are recorded in Tab. IV.

TABLE IV
COVERAGE OF DIFFERENT COMMUNICATION RADIUS

Radius	5m	6m	7m	8m	9m	10m
PSO	0.4611	0.605	0.7265	0.8341	0.8996	0.9433
FA	0.4637	0.6234	0.7537	0.8463	0.9145	0.9587
PFA	0.4688	0.6301	0.7597	0.8513	0.9202	0.9628
CFA	0.4697	0.6323	0.7632	0.8534	0.9266	0.9683
MCPCFA-B	0.4952	0.6672	0.7988	0.8931	0.9562	0.9881
MCPCFA-A	0.4891	0.6432	0.7787	0.8762	0.9398	0.9766
MCPCFA-R	0.4897	0.6587	0.7853	0.8861	0.9437	0.9802

It can be seen from the experimental results in Tab. IV that the coverage increases with the increase of communication radius. Through comparison, it can be seen that the performance of MCPCFA-B is the best. Compared with traditional FA, when the communication radius is 7m, the coverage optimized by MCPCFA family algorithm is improved by 4.51%, 2.5% and 3.16% respectively. When the communication radius is other values, the trend of the results is also consistent. Therefore, the effectiveness of mcpcfa proposed in this study and the applicability of this method in solving the WSN coverage optimization problem can be more determined.

VI. CONCLUSION

Aiming at the defects of traditional FA, MCPFA family algorithm is proposed in this paper to improve the problem that the algorithm is easy to fall into local optimal and run slowly. Meanwhile, due to the high time complexity of MCPFA family algorithm, this paper proposes MCPCFA family algorithm by introducing compact optimization method. Through the test experiments of several classical test functions in CEC2013, MCPCFA family algorithm shows superior performance compared with other methods. In the study of WSN, it is very important to maximize network coverage by effective node deployment under limited conditions. In order to solve this problem, the proposed MCPCFA family algorithm is applied to the 3D coverage research of WSN in this paper. The applicability of the proposed algorithm in this field is verified by simulation.

REFERENCES

[1] X. S. Yang, and X. He, "Firefly algorithm: recent advances and applications," International journal of swarm intelligence, vol. 1, pp. 36–50, 2013.

[2] Q. Ma, Z. Cao, and X. Zheng, "BOND: Exploring hidden bottleneck nodes in large-scale wireless sensor networks," ACM Transactions on Sensor Networks (TOSN), vol. 17, pp. 1–21, 2021.

[3] P. Bezerra, P. Y. Chen, J. A. McCann, and W. Yu, "Adaptive monitor placement for near real-time node failure localisation in wireless sensor networks," ACM Transactions on Sensor Networks (TOSN), vol. 18, pp. 1–41, 2021.

[4] X. Sui, S. C. Chu, J. S. Pan, and H. Luo, "Parallel Compact Differential Evolution for Optimization Applied to Image Segmentation," Appl. Sci., vol. 10, pp. 2195, 2020.

[5] M. Zhu, S. C. Chu, Q. Y. Yang, W. Li, and J. S. Pan, "Compact sine cosine algorithm with multigroup and multistrategy for dispatching system of public transit vehicles," Journal of Advanced Transportation, 2021.

[6] J. S. Pan, P. C. Song, S. C. Chu, and Y. J. Peng, "Improved Compact Cuckoo Search Algorithm Applied to Location of Drone Logistics Hub," Mathematics, vol. 8, pp. 3–33, 2020.

[7] A. Q. Tian, S. C. Chu, J. S. Pan, H. Q. Cui, and W. M. Zheng, "A Compact Pigeon-Inspired Optimization for Maximum Short-Term Generation Mode in Cascade Hydroelectric Power Station," Sustainability, vol. 12, 2020.

[8] J. P. Li, M. Gao, J. S. Pan, and S. C. Chu, "A parallel compact cat swarm optimization and its application in DV-Hop node localization for wireless sensor network," Wireless Networks, vol. 27, pp. 2081–2101, 2021.

[9] S. Karimi-Bidhendi, J. Guo, H. Jafarkhani, and S. C. Chu, "Energy-Efficient Node Deployment in Heterogeneous Two-Tier Wireless Sensor Networks With Limited Communication Range," IEEE Transactions on Wireless Communications, vol. 20, pp. 40–55, 2021.

Self-Adaptive Task Allocation for Decentralized Deep Learning in Heterogeneous Environments

1st Yongyue Chao
Institute of Automation, Chinese Academy of Sciences
Beijing, China
chaoyongyue2020@ia.ac.cn

2nd Mingxue Liao
Institute of Automation, Chinese Academy of Sciences
Beijing, China
mingxue.liao@ia.ac.cn

3rd Jiaxin Gao
Institute of Automation, Chinese Academy of Sciences
Beijing, China
jiaxin.gao@ia.ac.cn

4th Guangyao Li
Institute of Automation, Chinese Academy of Sciences
Beijing, China
liguangyao2020@ia.ac.cn

***Abstract*—The demand for large-scale deep learning is increasing, and distributed training is the current mainstream solution. Decentralized algorithms are widely used in distributed training. However, in a heterogeneous environment, each worker computes the same amount of training data resulting in a lot of wasted time for waiting the straggler. In this paper, we proposed a self-adaptive task allocation algorithm (SATA) which allows that each worker acquires the amount of training data adaptively based on the performance of workers in the heterogeneous environment. In order to show the applicability of SATA in heterogeneous clusters better, we set up the heterogeneous cluster composed of two or three different types of GPUs. Besides, we conduct a series of experiments to show the performance of SATA. The experimental results illustrate several advantages of SATA. SATA can accelerate distributed training about 3.3X that of All-Reduce as well as about 1.9X-3.8X that of AD-PSGD algorithm. And the total training time of SATA is reduced from 20 to 40 percentage compared to All-Reduce.**

***Index Terms*—distributed training, task allocation, All-Reduce, heterogeneity**

I. Introduction

As the state-of-the-art artificial intelligent approach, Deep Neural Networks (DNNs) play an important role in various applications, such as natural language process and computer vision. The increasing number of large-scale DNN model parameters provide high accuracy for complicated problems. However, training large-scale DNNs takes a long time and requires a large amount of memory, which is hard to train by single machines. Therefore, distributed training is the most popular method to alleviate the problem.

In distributed training, the model is trained by multiple workers running on a GPU cluster. The most common idea is data parallelism [1] in distributed training. All training data are assigned to workers for computing gradients and then workers aggregate gradients among others. Centralized and decentralized training with stochastic gradient descent (SGD) are the main approaches of data parallelism. One of the centralized approaches, Parameter Server (PS) [2], uses several nodes as servers, which collect the gradients from workers as well as update the model. However, this leads to inevitable communication bottleneck problems. Decentralized approaches resolve these difficulties based on All-Reduce [3]. Each worker only communicates with its neighbors to obtain the average model.

Although these approaches achieve wonderful results in homogeneous environments, the straggler problem still appear in heterogeneous environments. During the process of model training, each worker holds the same amount of training data and needs to wait for others aggregating gradients or parameters, so the whole training speed depends on the slowest worker in heterogeneous environments which contain different types of GPUs and different network bandwidths in distributed training. These heterogeneous environments result in slowing down the training speed of clusters with the straggler. In order to eliminate the effect of the straggler, many studies update the model based on asynchronous SGD instead of synchronous SGD. However, asynchronous SGD has lower accuracy and lower usage in distributed training. Another way is assigning tasks to each worker reasonably. The most common method for assigning tasks is to adjust mini-batch size of each worker and reassign dataset at every epoch. Some papers study the algorithms in self-adaptive task allocation. However, most of them are contributed to centralized training rather than decentralization based on the rough empirical speculations.

In this paper, we proposed a self-adaptive task allocation algorithm (SATA) for decentralized training. We implement our algorithm based on Ring All-Reduce. On the basis of algorithm, the amount of mini-batch data is balanced adaptively only by training information. We establish a mathematical model to describe the ratio of mini-batch size to global batch size in the current epoch for task allocation. To show the algorithm, we implement SATA based on the Ring All-Reduce algorithm in experiments. We set a series of experiments to show the improvement of training speed. Besides, we train a simple convolutional neural network model on MNIST dataset as well as some complex models including ResNet and VGG on CIFAR10 dataset. Experimental results show that self-adaptive task allocation algorithm can reduce 20

DOI reference number: 10.18293/SEKE2022-071

to 40 pecentage of training time compared to All-Reduce. And SATA can accelerate distributed training about 3.3X that of All-Reduce. Besides, SATA make progress in speedup compared to other algorithms.

II. Related Work

The straggler problem is very common in heterogeneous environments. It occurs due to the performance difference among workers and the discrepancy of communication speed and bandwidth. Existing efforts on the straggler problem can be classified into two types: algorithms based on task allocation and algorithms based on model averaging.

For algorithms based on adaptive task allocation, Yang et al. [4] proposed a batch orchestration algorithm, which balances the amount of mini-batch data according to the speed of workers. It makes a linear regression on training time and mini-batch size so that allocates tasks for workers based on slope changing. The weakness is that it introduces redundant training information to balance tasks. FlexRR [5] addresses the straggler problem by integrating flexible consistency bounds with temporary peer-to-peer work reassignment. FlexRR increases the cost of training due to extra communication in worker computing. The current studies are mostly apply for PS and based on empirical speculations.

For algorithms based on model averaging, countermeasures for synchronization are utilized, including asynchronous execution, bounded staleness, backup workers, adjusting the learning rate of stale gradients and so on. AD-PSGD [6], Partial All-Reduce [7] and gossip SGP [8] improve global synchronization with partial random synchronization. Chen et al. [9] proposed to set backup workers in the cluster, which allow gradient aggregations without all workers participation. DYNSGD [10] can dynamically adjust the local learning rate of worker according to the delay of the worker. Zhang et al. [11]has a similar idea, it adjusts the learning rate by the state of gradient. However, These approaches use the same amount of training data on every worker. They still waste the waiting time and resources of workers while synchronizing due to unreasonable training data distribution. Therefore, self-adaptive task allocation algorithm is urgently needed.

III. Method

In order to accelerate, we proposed a self-adaptive task allocation algorithm (SATA) by balancing the number of tasks among workers automatically. In SATA, we reassign training data as n subsets to n workers based on training time and the amount of subsets from the previous epoch. First, we establish a mathematical model coordinating with information of the previous epoch and the current epoch to allocate tasks. Then we integrate the All-Reduce algorithm to build our adaptive distributed training process.

A. task allocation model

In this subsection, we construct a function describing the amount of tasks which is only related to how long each worker takes to process and how much training data each worker holds at the last epoch. The detailed induction procedures are described below.

1) preliminaries: In order to describe the model better, we make the following premises and notations based on the real situations.

First, due to synchronization operations before aggregating local gradients, it is approximately though that all workers start and end at the same time in the process of each gradient aggregation with All-Reduce. Therefore, the gradient aggregation time of all workers t_c^i is equal. We set:

$$t_c^1 = t_c^2 = \cdots = t_c^n \tag{1}$$

Second, in synchronous SGD algorithms, total training procedure includes three steps: computation, synchronization and update. All workers execute the same preprocessing steps as well as they are blocked at barrier finally. Therefore, it can be approximately thought that total training time of all workers T_i will be equal. We set:

$$T_1 = T_2 = \cdots = T_n \tag{2}$$

Third, we set the ratio of local mini-batch size to global batch size in the previous epoch w_i^{k-1} and in the current epoch w_i^k. To avoid modifying learning rate along with the global batch size, we assume that global batch size keeps as a constant. Therefore, the sum of the ratio w and the changed ratio u_i will be set as:

$$w_1^k + w_2^k + \cdots + w_n^k = 1 \tag{3}$$

$$u_1 + u_2 + \cdots + u_n = 0 \tag{4}$$

Finally, to ensure the convergence of DNN model, the data subset D_i on worker i should be the uniform distribution over the assigned data samples. We distribute training data by sequential assignment on a sample-by-sample basis.

Besides, there are some parameters of the task allocation model to illustrate: gradient computing time t_s^i, synchronization waiting time t_w^i and gradient computing speed v_i where $v_i = D_i/t_s^i$, $T_i = t_s^i + t_w^i + t_c^i$ and the gap between synchronization waiting time is $\Delta t_w^{ij} = |t_w^i - t_w^j|$.

2) mathematical model: First, our objective functions are described as following:

$$\min_D \sum_{i=1}^{n} \sum_{j \neq i}^{n} \Delta t_w^{ij} \tag{5}$$

$$\min_D \quad T_1, T_2, \cdots, T_n \tag{6}$$

(5) and (6) are to minimize the total synchronization waiting time Δt_w^{ij} and total training time T_i. According to the former preliminaries (1) - (4), Δt_w^{ij} can be optimized to $\Delta t_w^{ij} = t_w^i - t_w^j = t_s^j - t_s^i = \frac{D_j}{v_j} - \frac{D_i}{v_i} = 0$. Finally we get the following formula:

$$\frac{Dw_j}{v_j} - \frac{Dw_i}{v_i} = 0 \tag{7}$$

In appendix, we computed the updated w_i^{k+1} in each epoch:

$$w_i^{(k+1)} = u_i + w_i^{(k)} = \frac{w_i^{(k)}/t_s^i}{\sum_{i=1}^{n} w_i^{(k)}/t_s^i} \tag{8}$$

It is found that the ratio w_i^{k+1} of the next epoch only depends on the ratio w_i^k of the current epoch and total computing time t_s^i. To ensure the feasibility of the model, each worker need to broadcast the above two parameters. In Fig.1 it shows the process of task allocation model in epochs. After getting the ratio w_i^{k+1} by the model, we design a complete algorithm called SATA, which shows the whole process of distributed training with the mathematical model.

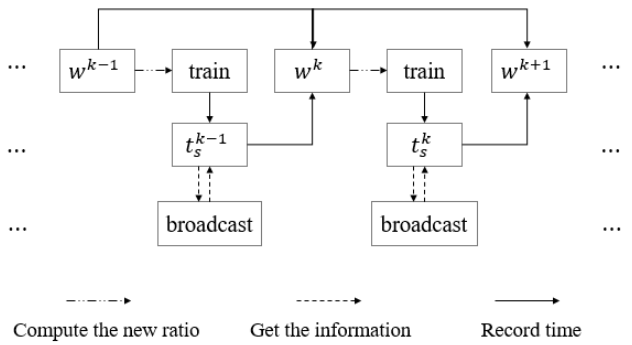


Fig. 1. The execution process of self-adaptive task allocation model, the ratio w_i^{k+1} depends on w_i^k and t_s

B. self-adaptive task allocation algorithm

In this subsection, we proposed a self-adaptive task allocation algorithm (SATA) to accelerate decentralized training. Before every epoch, SATA will call the model (8) to obtain the ratio w_i^k. Then SATA reassigns data subsets for all workers. Besides, local mini-batch size is revised by the product of global batch size N and the ratio w_i^k. As can be seen in Fig. 2, there are two cycles in the procedure of SATA. The outer cycle means reassigning data subsets for all workers based on the their own w_i^k at every epoch. The inner cycle means mini-batch data gradient aggregations in every epoch.

To accelerate, SATA makes use of self-adaptive task allocation to speed up. Fast workers take a long time to compute and stragglers take a short time, so the waiting time will be reduced naturally. Superior to other task allocation methods, SATA acquires the state of workers precisely based on reasonable derivation instead of empirical speculations. SATA can guarantee model parameters changing along with the gradient descent direction and converging to the stable loss. There is no changing in synchronous SGD back propagation. Therefore, the convergence point is equal to the original synchronous SGD. Besides, many papers([4,5,7,9,11]) also evaluate and derive the convergence of model when the amount of tasks is changed.

We describe the procedure of our SATA in Algorithm 1. There are three main parts in the SATA algorithm, determining the ratio, redistributing the dataset and training. Under relatively steady environment, w will be steady after several epochs. Therefore, the time of redistribution will be eliminated further in such an environment. By SATA algorithm, we can accelerate the whole procedure of distributed training.

Algorithm 1 Self-Adaptive Task Allocation (SATA) algorithm

Require:
Randomly initialize the data subset ratio for each worker i at epoch 0: w_i^0
Initialize gradient computing time: $t_s^i \leftarrow 0$
for the kth epoch $\in 1, 2, \cdots, N$ **do**
  Broadcast t_s^i and w_i^{k-1}
  Update w^k by Adaptive task allocation model
  if $w^k \neq w^{k-1}$ **then**
    Redistribute dataset to workers with w^k
  end if
  while data are not completely consumed **do**
    mini-batch train model
    Record and update (t_s^i)
    All-Reduce on model
  end while

IV. Experiments

In this section, we developed a series of experiments on self-adaptive task allocation algorithm to observe the DNN model training acceleration. First, we set up several experiments to show that SATA can speed up and be heterogeneity-tolerant on synchronization. We set a real heterogeneous cluster configured with different types of GPUs. Then We compare the performance between a homogeneous cluster and the heterogeneous one. Finally, we compare the results of SATA with other algorithms contributing in straggler problems. The results of all of these experiments show that SATA is suitable for complex heterogeneous environments.

A. acceleration by SATA algorithm in heterogeneous environments

We do experiments on multiple machines with multiple GPUs to evaluate results of SATA algorithm. We train ResNet, VGG and ConvNet models on three nodes as well as one node with multiple GPUs with different initial values of w. We record the ratio w, gradient computing time t_s and total training time T in each epoch. In Fig. 3 and 4, the total training time is reduced along with the increasing of epoch (subgraph c,f in Fig. 3 and 4). The gap between gradient computing time of two workers becomes smaller too (subgraph a,d in Fig. 3 and 4). After several epochs, the ratio w becomes steady (subgraph b,e in Fig. 3 and 4).

Further we compared results using the same ratio with the self-adaptive ratio among three groups of GPU clusters. The results in Fig. 5 show that with our SATA algorithm the training time will be reduced in heterogeneous environments.

B. compared to other algorithms for straggler problems

In the final experiment, we focus on the straggler problem comparing with All-Reduce and AD-PSGD algorithm. We set a straggler with 2X, 5X and 10X slowdown and set the speedup ratio of PS as 1 as done in Prague [7]. As shown in Fig. 6, our SATA algorithm converges more quickly compared

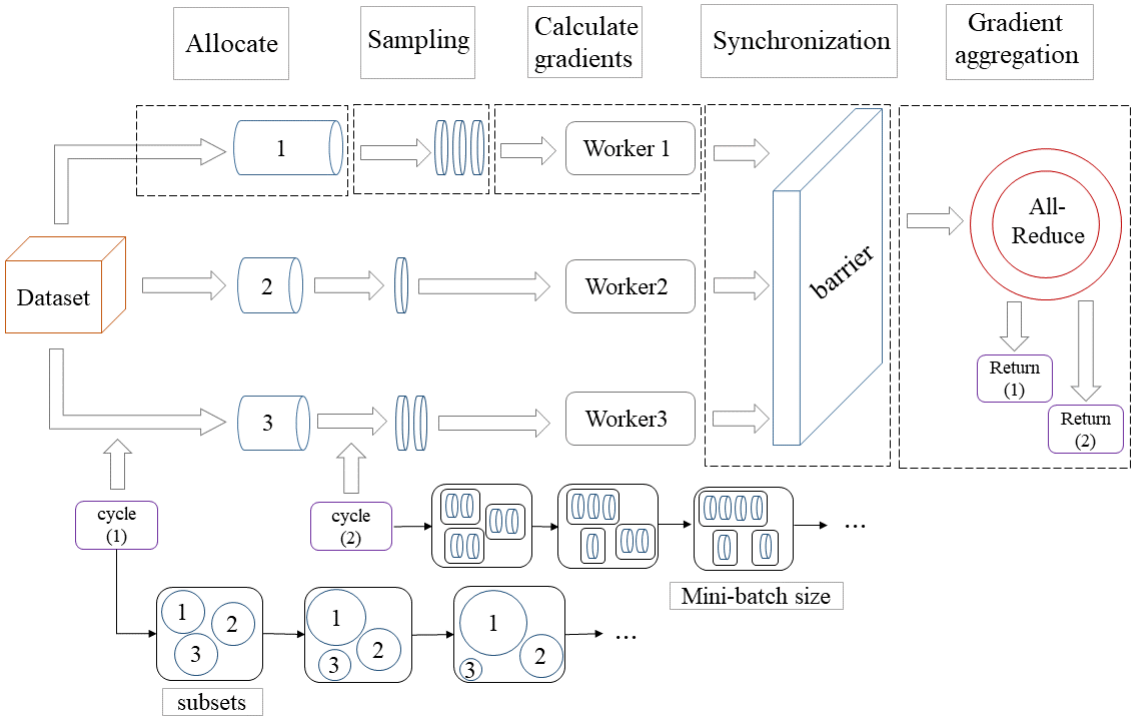


Fig. 2. The procedure of self-adaptive task allocation algorithm with three GPUs. Two cycles represent how the subsets and mini-batch sizes change as epoch increases. After one epoch, results return to cycle (1). After one gradient aggregation, results return to cycle (2)

to All-Reduce and AD-PSGD. In Fig. 7, SATA can reach about 3.3X that of All-Reduce given 2X/5X slowdown, 3.8X that of AD-PSGD under 2X slowdown and 1.9X under 5X slowdown.

V. Conclusion

To deal with stragglers in heterogeneous environments, we proposed a self-adaptive task allocation algorithm (SATA) in this paper. We firstly build a strict mathematical model for self-adaptive task allocation to determine the precise amount of mini-batch size iteratively updating for each worker. And then we give a detail process of our SATA algorithm for distributed training. By this algorithm, workers can reallocate local dataset adaptively according to current gradient computing time. We also set up a heterogeneous environment and designed a series of experiments to evaluate the performance of SATA for straggler problems. The experimental results show that SATA can accelerate distributed training about 3.3X that of All-Reduce and about 1.9X-3.8X that of AD-PSGD algorithm. It means that SATA has sound performance for straggler problems in heterogeneous environments.

In future, we will focus on distributed self-adaptive task allocation algorithms which will eliminate broadcast or centralized communication.

References

[1] T. Ben-Nun and T. Hoefler. “Demystifying Parallel andDistributed Deep Learning: An In-Depth Concurrency Analysis”. In: ACM Computing Surveys 52.4 (2018)

[2] Mu Li, Zhou Li , Alex Smola, Parameter server for distributed machine learning, In NIPS, 2013

[3] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaim-ing He. Accurate, large minibatch SGD: training imagenet in 1 hour.CoRR, abs/1706.02677, 2013

[4] E. Y ang, D. K. Kang, and C. H. Y oun. “BOA: batch orchestration algorithm for straggler mitigation of distributed DL training in heterogeneous GPU cluster”. In: The Journal of Supercomputing (2019).

[5] Aaron Harlap et al. “Addressing the straggler problem for iterative convergent parallel ML”. In: Proceedings of the Seventh ACM Symposium on Cloud Computing, Santa Clara, CA, USA, October 5-7, 2016.

[6] X. Lian et al. “Asynchronous Decentralized Parallel Stochastic Gradient Descent”. In Proceedings of the 35th International Conference on Machine Learning, PMLR 80:3043-3052, 2018(2017).

[7] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. 2020. Prague: High-Performance Heterogeneity-Aware Asynchronous Decentralized Training. Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. Association for Computing Machinery, New York, NY, USA, 401–416.

[8] Yeo S , Bae M , Jeong M , et al. Crossover-SGD: A gossip-based communication in distributed deep learning for alleviating large mini-batch problem and enhancing scalability. 2020.

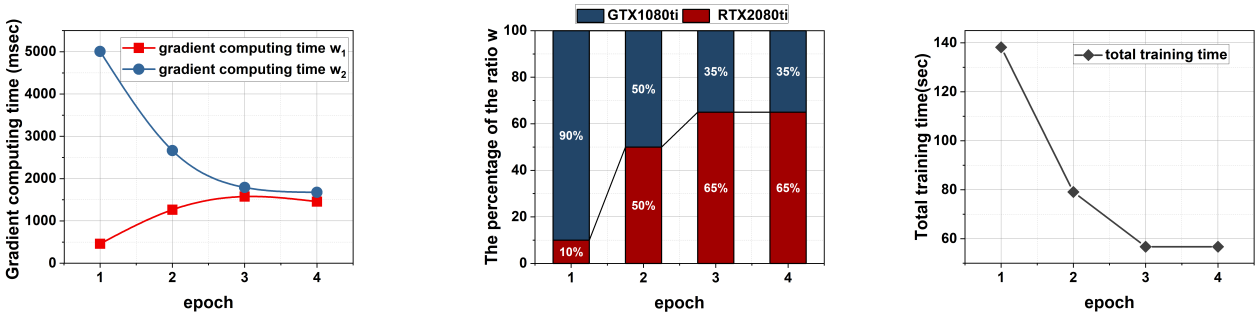
[9] Chen, Jianmin et al. “Revisiting Distributed Synchronous SGD.” ArXiv abs/1702.05800 (2016): n. pag.

[10] J. Jiang et al. “Heterogeneity-aware Distributed Parameter Servers”. In: Acm International Conference. 2017,pp. 463–478.

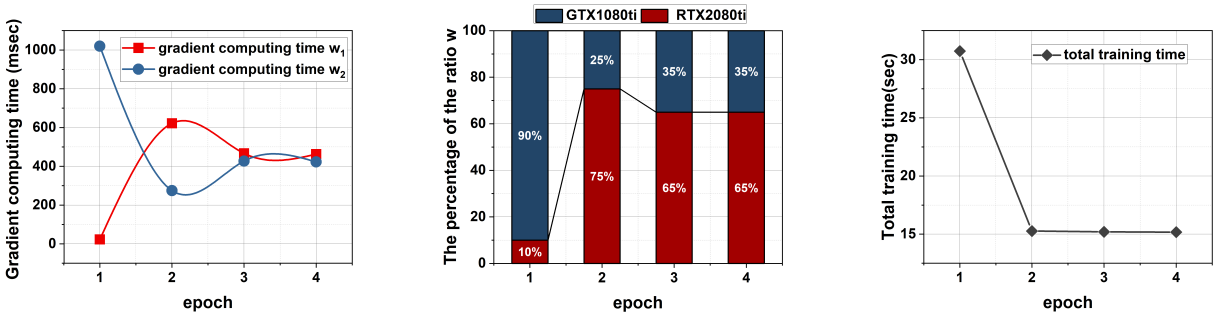
[11] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. 2016. Staleness-aware async-SGD for distributed deep learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI’16). AAAI Press, 2350–2356

[12] C. Szegedy et al. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”. In: AAAI (2016)

[13] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”.In: Computer Science (2014).

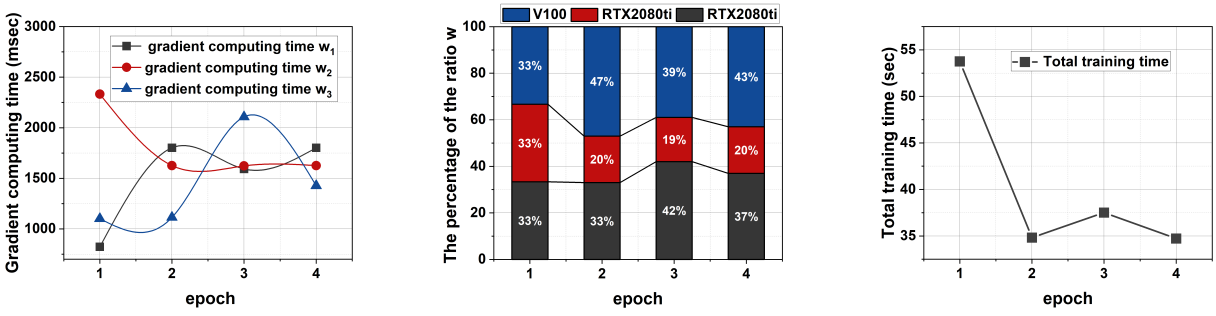


(a) ResNet50-t_s (b) ResNet50-W (c) ResNet50-T

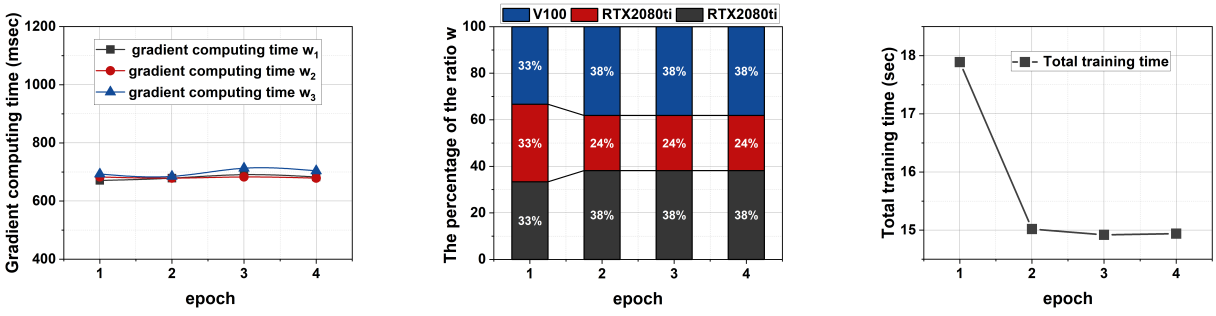


(d) VGG16-t_s (e) VGG16-W (f) VGG16-T

Fig. 3. the gradient computing time before one gradient aggregation, the ratio and training time from one machines with RTX2080ti and GTX1080ti.



(a) ResNet50-t_s (b) ResNet50-W (c) ResNet50-T



(d) VGG19-t_s (e) VGG19-W (f) VGG19-T

Fig. 4. the gradient computing time before one gradient aggregation, the ratio and training time from three machines with 2*RTX2080ti and V100 respectively. (a) and (d) represent gradient computing time among workers. (b) and (e) represent the ratio w_i^k. (c) and (f) represent the total training time

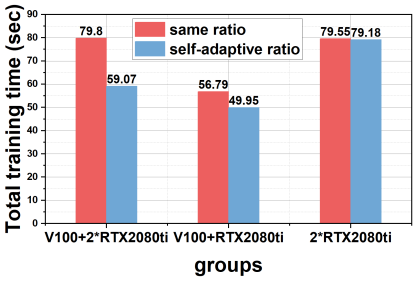
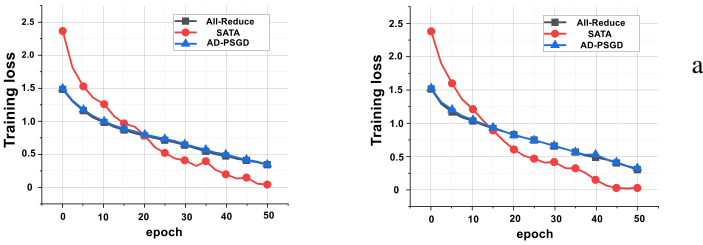


Fig. 5. total training time in one epoch compared with different types of GPUs



(a) convergence curve in 2X slowdown (b) convergence curve in 10X slowdown

Fig. 6. the training convergence curve of models for ResNet50.

APPENDIX

Assuming that the kth epoch task allocation of workers is $w_i^{(k)}$, the $(k+1)$th epoch is $w_i^{(k+1)}$, and the changed ratio is u_i as (9)

$$w_i^{(k+1)} = w_i^{(k)} + u_i, i \in [1..n] \tag{9}$$

Due to the waiting time of n workers expected to be 0, so the relationship between workers can be expressed as (10)

$$\frac{Dw_i^{(k+1)}}{v_i} - \frac{Dw_j^{(k+1)}}{v_j} = 0, i \neq j \tag{10}$$

From the view of the linear equation system, (10) is equivalent to (11).

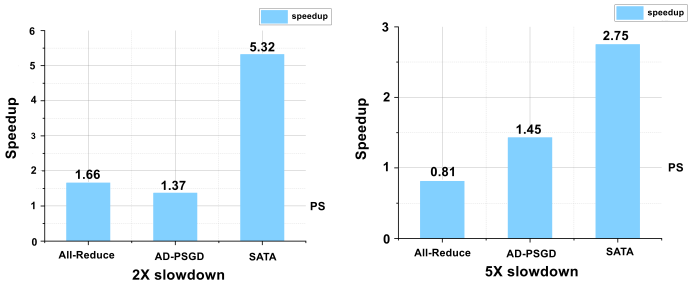
$$\frac{Dw_i^{(k+1)}}{v_i} - \frac{Dw_j^{(k+1)}}{v_j} = 0, \forall i, j = (i+1) mod\ n \tag{11}$$

Simply to get:

$$\frac{w_i^{(k)} + u_i}{v_i} - \frac{w_j^{(k)} + u_j}{v_j} = 0, \forall i, j = (i+1) mod\ n \tag{12}$$

Extract the coefficient matrix to get

$$A' = \begin{bmatrix} \frac{1}{v_1} & \frac{-1}{v_2} & 0 & \cdots & \cdots & 0 & 0 \\ 0 & \frac{1}{v_2} & \frac{-1}{v_3} & \cdots & \cdots & 0 & 0 \\ 0 & 0 & \frac{1}{v_3} & \frac{-1}{v_4} & \cdots & \cdots & 0 \\ & & & \vdots & & & \\ 0 & 0 & \cdots & \cdots & 0 & \frac{1}{v_{n-1}} & \frac{-1}{v_n} \end{bmatrix} \tag{13}$$



(a) speedup in 2X slowdown (b) speedup in 5X slowdown

Fig. 7. the training speedup of models

The global batchsize remains unchanged, so we have (14) and then (15).

$$w_1 + w_2 + \cdots\cdots + w_n = 1 \tag{14}$$

$$u_1 + u_2 + \cdots\cdots + u_n = 0 \tag{15}$$

Combining (13) with (15) we have (16)

$$A = \begin{bmatrix} & A' & & \\ 1 & 1 & \cdots & \cdots & 1 & 1 \end{bmatrix} \tag{16}$$

The constant term is

$$b = \begin{bmatrix} \frac{w_2^{(k)}}{v_2} - \frac{w_1^{(k)}}{v_1} \\ \frac{w_3^{(k)}}{v_3} - \frac{w_2^{(k)}}{v_2} \\ \vdots \\ \frac{w_n^{(k)}}{v_n} - \frac{w_{n-1}^{(k)}}{v_{n-1}} \\ 0 \end{bmatrix} \tag{17}$$

Therefore we only need to solve (18).

$$\begin{aligned} \mathrm{A} \bullet \mathrm{u} &= \mathrm{b} \\ u &= [u_1, u_2, \cdots\cdots, u_n]^T \end{aligned} \tag{18}$$

Finally we get the solution in terms of u as (19).

$$u = \frac{v_i}{\sum_{j=1}^{n} v_j} - w_i^{(k)} \tag{19}$$

A Federated Model Personalisation Method Based on Sparsity Representation and Clustering

Hailin Yang, Yanhong Huang*, Jianqi Shi and Fangda Cai*

National Trusted Embedded Software Engineering Technology Research Center,
East China Normal University, Shanghai, China
hlyang@stu.ecnu.edu.cn
{yhhuang, jqshi, fdcai}@sei.ecnu.edu.cn

***Abstract*—As federated learning (FL) becomes more extensively employed, it attracts an increasing number of scholars and practitioners. In contrast to traditional decentralized machine learning approaches that acquire users' raw data, FL gathers locally updated gradients, protecting their privacy. However, different users may have disparate data distributions, resulting in underperformance of the federated model. It is beneficial to adapt the federated model to various data distributions. Numerous personalisation approaches have been examined, but most of them are limited to a single device with minimal data, making them susceptible to bias and overfitting. In fact, the data distributions of certain users are similar, and these similarities can be leveraged to increase the efficacy of personalisation. In this research, we describe a sparsity-based clustering method, as well as a federated personalisation strategy based on it. Our method mitigates the impact of non-IID data and generates more accurate local models. The trials reveal that it outperforms several of its counterparts.**

***Index Terms*—Federated Learning, Privacy, Personalisation, Clustering**

I. Introduction

Increasingly, internet-connected devices are becoming an integral part of people's daily life. During their operation, they create a significant amount of data. Alough this data is of immense financial worth, most users don't want their privacy to be abused. It's meaningful to exploit local resources to collaboratively train machine learning (ML) models while keeping data on the devices. Federated learning [1] satisfies this requirement for it only requires participants to upload their locally determined gardiends to a sever and then aggregate them. The most well-known aggregation algorithm is the Federated Average (FedAvg), which simply averages each user's gradients. And vanilla FL refers to FL using FedAvg. However, the global model generated by vanilla FL is usually not satisfying for a specific user, because data is usually Non-independently and identically distributed(non-IDD) between clients [2].

A wealth of ways have been presented to solve the difficulties that data heterogeneity poses for FL. The global model can be improved by, for instance, retraining its parameters using input from a single user to produce a unique local model [3]. Yu et al. [4] introduced two further ways of personalization, namely multi-task learning and knowledge distillation. In addition, Arivazhagan et al. [5] suggested a federated learning strategy with personalization layers.

While all of these tactics contribute to personalisation, they all suffer from the same disadvantage: insufficient data. As a result, scholars investigated the feasibility of using federated approaches to indirectly augment the data used for personalisation. Liu et al. [7] describes a method that groups clients based on their data distribution, and use FedAvg within clusters to personalize the global model. Nonetheless, it was assumed that clients belonging to the same cluster would have the same data distribution, which occurs infrequently.

Clients' data distributions are similar but not identical, i.e., clients' preferences are consistent in general but varied in specifics. According to Liu et al. [7] the sparsity of an input image's feature map is often unique and may be utilized as a representation of the image. Simultaneously, this form of data representation is privacy-protective, as it conceals the underlying data. We have taken this idea and created some beneficial changes.

In this research, we offer a privacy-preserving federated personalisation technique to mitigates non-IID impact of FL. To begin, we transform the data distribution of each client into a sparsity vector. Then, we cluster these sparsity vectors to achieve client clustering. Following that, federated personalisation occurs within each cluster.

Our technique implemented federated learning at a finer grain, allowing clients in the same cluster to exchange critical knowledge without sacrificing their identity. Each client can obtain a model developed specifically for them. This work's contribution can be summarized as follows:

- We propose a privacy-preserving clustering method based on vector representations of clients' data distributions.
- Our strategy allows for some variation in the data distribution of clients within the same cluster, which is more practical.
- To maximize the effect of personalisation, we innovatively employ a FL approach with personalisation layers within clusters.
- We ran multiple relevant experiments and found that our technique outperforms its competitors.

*Corresponding Author
DOI reference number: 10.18293/SEKE2022-100

The remainder is structured as follows: Section II provides background information. Section III elaborates on the proposed method. Section IV details the experiments and their outcomes. Additionally, Section V discusses related work. Section VI concludes and suggests some potential directions worth pursuing.

II. Preliminaries

As a starting point, this section gives background information on the concepts of sparsity, clustering, and personalisation in order to aid in the comprehension of the technique we are proposing.

A. Sparsity

Sparsity is a numerical value that indicates the percentage of zeros in a matrix. It is frequently used to accelerate the inference process of Convolutional Neural Networks (CNNs) [10]. The first instance of channel-wise sparsity being used for data representation has appeared in [7]. A sparsity vector is constructed as follows, v_k represents the sparsity of the k_{th} channel.

$$V = \{v_1, v_2, v_3, v_4, ..., v_{k-1}, v_k\}$$

While feature maps are frequently used to portray raw data, they reveal too much fundamental information and are vulnerable to privacy leaks. In comparison, it is nearly impossible to deduce sufficient knowledge from a sparsity vector, offering a higher level of privacy protection. On the other hand, this representation format also contributes to the reduction of communication and processing overhead, as a sparsity vector is a number that can be transferred and calculated rapidly.

B. Clustering

Clustering is a technique for grouping similar objects together, with the core issue being determining how to measure similarity. There are a variety of distances that can be used to estimate similarity. Among these, Euclidean Distance [10] is a straightforward yet widely used one. The Euclidean distance between two vectors is calculated as follows:

$$Dist(V_a, V_b) = \sqrt{\sum_{j=1}^{j=k} (V_a^{(j)} - V_b^{(j)})^2} \quad (1)$$

V_a and V_b denote vectors, while $V_a^{(j)}$ and $V_b^{(j)}$ denote the j^{th} component of V_a and V_b, respectively.

Kmeans is a well-known technique for clustering data based on Euclidean distance. It initially selects k points as centroids and then groups the points closest to the centroid with them. The cluster centroid is iteratively updated until the total distance between it and the points inside the cluster no longer decreases. At this point, the clustering results are obtained.

C. Personalisation

FL collects and aggregates local gradients from participants during each communication round in order to update the global model. When users' data distributions are comparable, the global model easily converges and demonstrates superior performance versus the local models. However, this is not always the case. Once data distribution is highly variable, the global model typically underperforms.

A bulk of academics have examined the difficulties inherent in applying federated learning to heterogeneous data sources. A frequently used method is to further personalize the global model. Personalisation, in general, refers to optimization techniques that make the federated global model better suited to local clients.

III. Approach

This section details our strategy. For starters, we demonstrate how to generate sparsity vectors from the user's raw data which consist of images with different classes. Following that, we describe the clustering procedure using sparsity vectors. Finally, we describe the federated cluster-wise personalisation technique.

A. Generation of sparsity vectors

Prior to constructing a sparsity vector for a client, we must first determine the sparsity of each image on that client. The sparsity of an image is computed as shown in Figure 1.

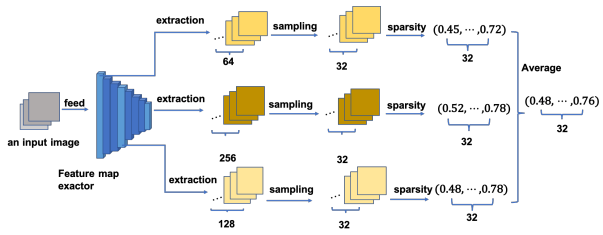


Fig. 1. The process of transforming a single image into a sparsity vector.

For feature extraction, we employ the CNN structure. CNNs are composed of multiple layers of neurons, with each layer producing a feature map. We calculate the sparsity of feature maps produced by Rectified Linear Unit (ReLU) layers because ReLU layers set a section of neurons' outputs to zero, resulting in sparse feature maps. The VGG [8] network

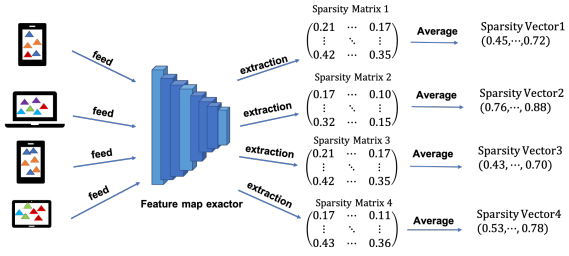


Fig. 2. All data are arranged in a matrix by their sparsity vectors, and the first dimension of this matrix is averaged to generate a vector representing the data distribution..

stacked small convolutional kernels instead of large ones repeatedly, increasing the network's depth while maintaining the same field of perception, and thereby improving the network's capacity to extract features. VGG-11 is the simpleset

VGG network. We replace the original fully connected layers of VGG-11 with a global average pooling layer, and then use it as the feature extractor. Feature maps become more representive with the modified VGG-11 for it establishes a direct connection between feature maps and classes.

Additionally, we take the first three ReLU layers as the extraction layers rather than a single one to enhance the extraction. Feature maps from multiple ReLU layers do not necessarily have the same number of channels, so we picked k common channels to calculate a sparsity vector.

As a result, the sparsity vectors from separate layers become identical in size and can be averaged into one. We will obtain a n∗k dimensional sparsity matrix for a user with n data. A one-dimensional sparsity vector with k elements is created by averaging the sparsity matrix over the first dimension. Figure 2 depicts the process of expressing clients' data as sparsity vectors.

B. Clustering Based on Sparsity Vectors

The concept of sparsity-based clustering lies at the heart of our approach. With respect to this idea, it was first put forward by Liu et al. [7]. According to them, when the distance between two sparsity vectors is smaller than a certain threshold, they are considered similar. However, it's a challenge to determine this crucial hyperparameter in practice. It can be counterproductive if the clustering result is inaccurate. Considering the clustering categories are less extensive than the threshold, we employ k-means clustering to obtain better results.

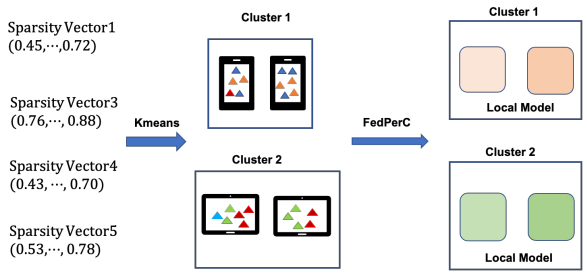


Fig. 3. Kmeans is performed based on sparsity vectors, and then FedPerC is conducted based on clusters.

The sparsity vector representing a client is computed using all its data. Then we compute the Euclidean Distance between each pair of vectors and then cluster them using the k-means algorithm. Based on the clustering results, we ran FedPerC to personalise the federated global model that was previously trained. This procedure is depicted in Figure 3. FedPerC is a cluster-based algorithm for federated personalization that will be talked about in the next section.

For clustering, k is critical. If k is set too high, clustering will be less successful, and if set too low, the benefits of federated learning will be missed. We can choose k according to our prior understanding of the dataset. If no prior knowledge exists, we can do several trials with randomly chosen clients to determine the best value of k and then apply it to all clients.

C. Federated Personalisation within Clusters

We expand personalisation from a single device to all devices inside a cluster, and we refer to this approach as FedPerC. To take use of the similarities in data distributions of users inside a cluster without losing sight of their differences, we split the model in two. The concept is derived from [5].

$W_{B,i}^{(k)}$ and $W_{P,i}^{(k)}$ signifiy the base layer and personalisation layer parameters of client i at the k_{th} training round, respectively. $W_{B,i}^{(k)}(W_{P,i}^{(k)})$ denote the entire model consist of this two part. LocalUpdate function refers to the training process on a single device. $W_{L,i}^{(k)}$ donate the local model of client i at the k_{th} training round. A formal description of FedPerC is given in Algorithm 1.

Algorithm 1 FedPerC

Input: D_i-the data of the i_{th} client
C-clusters of clients
W_g-federated global model
N-training rounds
Output: W_l-personalised local models

Initialisation: $W_l \leftarrow W_g$
for k = 0,1,2,...,N-1 **do**
  for c in C **do**
    for i in c **do**
      $W_{l,i}^{(k)} \leftarrow \text{LocalUpdate}(W_{l,i}^{(k-1)}, D_i)$
      $W_{B,i}^{(k)}(W_{P,i}^{(k)}) \leftarrow W_{l,i}^{(k)}$
    Aggregate: $W_{B,c}^{(k)} \leftarrow \sum_{i=1}^{c} W_{B,i}^{(k)}$
    for i in c **do**
      $W_{l,i}^{(k)} \leftarrow W_{B,c}^{(k)}(W_{P,i}^{(k)})$
return W_l to local clients

Only the weights of base layers must be uploaded to the parameter server throughout the personalization process, the weights of personalization layers remain local. In Figure 4, we illustrate the process of personalizing within clusters.

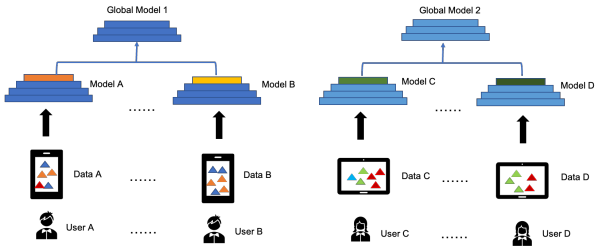


Fig. 4. Users within a cluster share the common base layers, and their personalisation layers can be adapted to individual data.

A formal description of the personalised goals is as follows: Suppose M_g donates the global model trained federally on data from all clients and M_l^i denotes the local model of the i^{th} client after personalisation. The number of clients is n, and C_i represents the cluster to which the i^{th} client belongs. The goal of FedPerC is to update M_g with the assistance of C_i

TABLE I
ACHIEVED ACCURACY (%) ON CIFAR-100 OF THE BASELINE AND DIFFERENT PERSONALIZATION METHODS USING RESNET34.

Method	client1	client2	client3	client4	client5	client6	client7	client8	client9	client10
BaseLine	45.17	41.39	43.18	41.11	41.67	35.83	38.33	39.72	38.06	41.39
FineTune	64.49	61.39	64.49	59.44	64.17	62.50	61.67	63.89	64.44	62.50
PFA	63.92	57.50	65.63	58.33	67.22	63.89	60.83	63.06	61.94	64.17
Ours	**65.34**	**65.28**	**65.06**	**63.06**	**68.89**	**65.56**	**63.89**	**64.72**	**65.28**	**65.56**
Method	client11	client12	client13	client14	client15	client16	client17	client18	client19	client20
BaseLine	37.78	41.94	40.56	38.06	34.44	43.06	39.17	44.47	41.76	41.76
FineTune	56.94	62.22	60.2	58.33	57.22	65.00	59.17	64.24	63.92	68.47
PFA	58.33	64.17	59.44	60.56	60.56	67.50	58.61	66.86	63.07	63.64
Ours	**61.39**	**65.83**	**61.39**	**62.50**	**61.67**	**67.50**	**63.61**	**70.35**	**66.19**	**70.17**

TABLE II
ACHIEVED ACCURACY (%) ON CIFAR-100 OF THE BASELINE AND DIFFERENT PERSONALIZATION METHODS USING MOBILENETV1.

Method	client1	client2	client3	client4	client5	client6	client7	client8	client9	client10
BaseLine	37.22	33.33	34.38	31.11	38.33	35.83	35.56	40.29	37.79	41.68
FineTune	66.48	61.39	62.22	61.11	66.39	65.57	66.11	60.56	63.33	63.89
PFA	65.63	62.78	63.35	61.11	65.56	66.39	64.72	63.89	66.39	60.00
Ours	**68.75**	**65.56**	**65.34**	**65.56**	**70.56**	**68.06**	**68.33**	**65.56**	**67.50**	**69.44**
Method	client11	client12	client13	client14	client15	client16	client17	client18	client19	client20
BaseLine	37.50	41.11	35.00	29.44	33.33	41.11	34.44	40.70	40.63	31.82
FineTune	58.33	63.33	60.28	56.11	58.61	62.78	57.50	64.54	60.51	63.64
PFA	60.56	57.22	59.17	53.61	57.50	63.33	60.56	67.73	60.23	64.21
Ours	**62.50**	**65.56**	**63.06**	**58.61**	**63.33**	**66.94**	**63.61**	**69.19**	**63.92**	**67.33**

and generate a personalised model M_l^i. The following is the objective function:

$$min \sum_{i=0}^{n} \mathcal{L}(D_i, C_i, M_g) \tag{2}$$

where $\mathcal{L}$ denotes the loss function, and in this work we use the cross-entropy loss.

IV. EXPERIMENT

Detailed descriptions of the experimental setup, including the datasets and models used, as well as the implementation details and methodologies for comparison, are provided in this part.

A. Settings

On two widely used network architectures, ResNet [11] and MobileNet [12], we conducted experiments to validate our technique. ResNet addresses the issue of gradient vanishing through a residual structure, improving performance. MobileNet pioneered the notion of depthwise separable convolution, drastically reducing model parameters and making itself suitable for low-resource mobile devices. We employed ResNet34 and MobileNetV1 on the CIFAR-10 and CIFAR-100 datasets [6], respectively. They are both image classification datasets, with 10 and 100 categories, respectively. Additionally, we implement our strategy in Python using PyTorch [13].

At first, we simulate a federated environment. To be more precise, we simulate 20 clients and 5 clients roughly belonging to a type of distribution for each dataset. For clients from CIFAR-10 that are expected to be drawn from the same distribution, 80% of their data came from two common categories and 20% from a random category. CIFAR-100 is processed similarly, with similar clients sharing twenty classes of data and holding an additional five classes of data. Thus, four distinct data distributions correspond to four clusters. Furthermore, to mimic the finite nature of client data, we limit the quantity of data available to each client to no more than 1000.

Experiments was conducted to determine the values of the hyperparameters. The final experimental parameters are as follows: 50 and 30 rounds were conducted respectively for training the federated model and personalizing. The batch size and local epochs are set to 64 and 4, while the learning rate and momentum are set to 0.01 and 0.5. The number of channels used for feature extraction is 32, and the indexes of the ReLU layers involved are 0, 1, 2. In federated personalization, the final fully connected layer serves as the personalization layer for both networks.

B. Results

Principal component analysis (PCA), a dimension reduction technique, was used to visualize the clustering. In Figure 5, we plot these reduced two-dimensional points to visualize the clustering, with different colors signifying different clusters. Figure 6 depicts the process of federally training a global

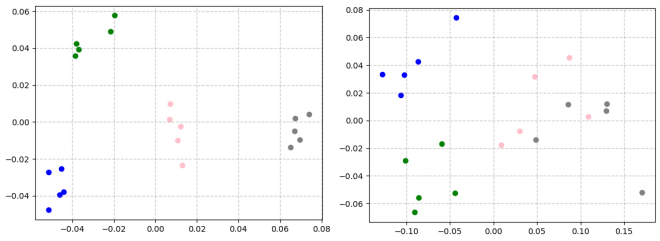


Fig. 5. Visualization of clustering on CIFAR-10(left) and CIFAR-100(right).

model on the CIFAR-100 dataset using ResNet34 and MobileNetV1. As can be observed, the heterogeneous distribution of the data causes the global model to over-fit, necessitating personalisation. It can be observed that each cluster contains 5

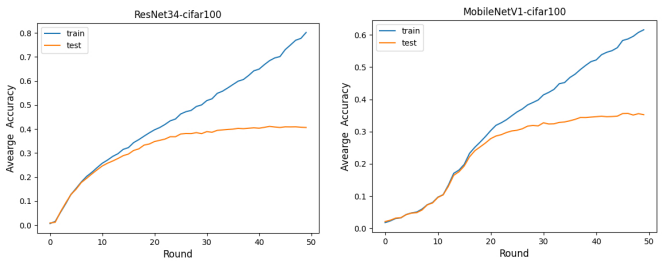


Fig. 6. The ResNet34 and MobileNetV1 training processes on CIFAR-100.

clients, and different clusters are spread in different positions in space. The clustering findings demonstrate that our approach matches our expectations.

we use the global model's test accuracy to establish a baseline for each client. The global model is then optimized using a variety of personalization methods, including finetune, PFA, and FedPerC. Fine-tuning is a popular migrating learning strategy that retrains all or part of the parameters. In this case, we refer to it as "retraining all parameters." And it represents the personalisation effect on a single device. PFA is a framework introduced in paper [7] that organizes clients first and then personalizes within groups with the FedAvg algorithm.

TABLE III
THE CLUSTER-LEVEL AVERAGE ACCURACY AND OVERALL AVERAGE ACCURACY OF RESNET34 ON CIFAR-100.

Method	cluster1	cluster2	cluster3	cluster4	Avg
BaseLine	42.50	38.67	38.56	42.04	40.44
FineTune	62.28	61.56	58.61	62.58	61.26
PFA	61.66	62.50	61.65	61.39	61.80
Ours	**65.52**	**65.00**	**62.56**	**67.56**	**65.16**

TABLE IV
THE CLUSTER-LEVEL AVERAGE ACCURACY AND OVERALL AVERAGE ACCURACY OF MOBILENETV1 ON CIFAR-100.

Method	cluster1	cluster2	cluster3	cluster4	Avg
BaseLine	34.87	38.23	35.27	37.74	36.53
FineTune	63.52	63.89	59.33	61.79	62.13
PFA	64.12	64.28	57.61	63.21	62.31
Ours	**67.15**	**67.78**	**62.61**	**66.20**	**65.94**

During the experiment, each approach is run three times, and the results of the best performing rounds are averaged as a reflection of the method's performance. Table I and Table II demonstrate experimental findings. When compared to alternative localization approaches, we can see that our approach provides a variable degree of accuracy enhancement to each client. In Table III and Table IV, we display the average accuracy within and across clusters. The findings indicate the superiority of our strategy.

TABLE V
THE IMPACT OF CLUSTERING ON RESNET34'S PERFORMANCE ON CIFAR-100.

Client Number	Random Clustering	No Clustering	Sparsity-based Clustering
client1	63.07	61.93	**65.34**
client2	60.83	63.06	**65.28**
client3	61.65	61.08	**65.06**
client4	61.11	60.28	**63.06**
client5	63.33	66.39	**68.89**
client6	60.83	59.72	**65.56**
client7	59.17	58.06	**63.89**
client8	58.61	61.39	**64.72**
client9	61.11	62.50	**65.28**
client10	62.22	61.94	**65.56**
client11	56.67	56.11	**61.39**
client12	61.11	61.67	**65.83**
client13	57.78	58.61	**61.39**
client14	54.44	60.00	**62.50**
client15	55.56	54.72	**61.67**
client16	64.17	65.28	**67.50**
client17	60.00	58.06	**63.61**
client18	61.92	63.66	**70.35**
client19	59.38	60.23	**66.19**
client20	63.07	63.35	**70.17**
Avg	60.30	60.90	**65.16**

Furthermore, to see whether clustering affects the impact of personalisation, we examined the experimental findings of ResNet34 on CIFAR-100 under three conditions: clustering, no clustering, random clustering. Table V displays detailed experimental data.

The experimental investigations indicate that random clustering has a comparable or perhaps slightly lower personalising impact than no clustering. When employing our clustering approach, both models increase their accuracy on CIFAR-100 by 5% roughly.

TABLE VI
CIFAR-10 RESULTS FOR THE BASELINE AND DIFFERENT PERSONALIZATION METHODS.

Model	Method	Avg
ResNet34	BaseLine	53.25
	FineTune	82.55
	PFA	79.45
	FedPerC	**84.48**
	FedPerC(Random Clustering)	82.55
	FedPerC(No Clustering)	81.88
MobileNetV1	BaseLine	51.28
	FineTune	83.53
	PFA	78.95
	FedPerC	**85.08**
	FedPerC(Random Clustering)	82.85
	FedPerC(No Clustering)	83.33

When the clustering is suitable, our technique is supposed to have a favorable impact since it takes into account the users' similarities and differences. However, when the clustering is inaccurate, it might be detrimental. When all clients are treated as belonging to a cluster, the result is identical to not clustering, and our technique degrades to the FedPer algorithm given in article [5].

Table VI presents the average accuracy of all previously mentioned personalisation techniques on CIFAR-10. The ac-

curacy boost on CIFAR-10 is not as significant as on CIFAR-100, most likely because CIFAR-100 has a higher number of classes and the variances across clients within a cluster are greater. Additionally, a user often owns various classes of images in reality, so our approach has its practicalities.

V. Related Work

Federated learning has extended the range of applications for artificial intelligence (AI). In recent years, it has become a popular research topic. However, It also confronts obstacles on a variety of fronts [2], including privacy breaches and heterogeneous data distribution.

Privacy violations may cause a great deal of grief in people's lives. As a result, individuals are becoming more conscious of their right to privacy. Furthermore, the implementation of applicable rules, like the General Data Protection Regulation (GDPR) [14], not only officially protects users' privacy but also drives adjustments in machine learning algorithms that need access to users' raw data.

Secure multiparty computing (SMC) [15] is a privacy-preserving technique that based on mathematical theory. It safeguards all parties' input data while generating accurate results, and is especially beneficial in the absence of a trustworthy third party. However, SMC requires substantial computer power and network resources for encryption and decryption, which is not achievable for resource-constrained devices.

Differential privacy (DP) [16] preserves privacy by introducing noise into the data to diminish its sensitivity. Noise must be provided to make it more difficult to derive information about users without significantly affecting the distribution of data. On the other hand, DP makes the data less reliable, which has an effect on how well the training works.

As a result, when the idea of federated learning was initially proposed, it sparked a great deal of attention. Most importantly, FL enables resource-constrained devices to cooperate together to train a shared model that benefits each device without requiring access to any device's raw data. When the distribution of data for separate clients is roughly the same, it proves to be a realistic technique [1]. Unfortunately, this is not always the case. Therefore, FL has established itself as a research center dedicated to tackling the issues raised by data non-IDD.

Mansour et al. [17] introduces a personalisation method based on clustering. However, users are required to give out some raw data in return for precise clustering results, compromising data privacy. Wang et al. [3] applies transfer learning to personalisation, retraining all or part of the parameters of the federated model on local data. Another one, based on the concept of transfer learning, is provided in [18], in which meta-learning is utilized to establish the global model and then fine-tuning approaches are employed to achieve personalisation. Yu et al. [4] recommended that networks be divided into two components: base layers and personalisation layers, with the former being trained by all users collectively and the latter by each user individually.

VI. Conclusion And Future Work

In this work, we propose an algorithm that implements personalisation by clustering users based on a privacy-protected representation of their original data. This is an exploratory way to addressing the non-IID dilemma of FL, although it has certain downsides. For example, though the sparsity patterns of various inputs are often distinct, there may be exceptions that result in incorrect clustering conclusions. How to enhance the logic of clustering is a topic that deserves to be investigated more in the future.

Since it is almost impossible to derive the original data from the sparsity representation, we regard it as privacy-preserving. However, this representation's ability to protect privacy has not been formally demonstrated. Additionally, incorporating established privacy-protection mechanisms such as DP is a point of improvement. On the other hand, our method is limited to CNNs, leaving the possibility of extending it to other network architectures for future implementation.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Artificial intelligence and statistics, 2017, pp. 1273–1282.

[2] P. Kairouz et al., "Advances and open problems in federated learning," arXiv preprint arXiv:1912.04977, 2019.

[3] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," arXiv preprint arXiv:1910.10252, 2019.

[4] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," arXiv preprint arXiv:2002.04758, 2020.

[5] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," arXiv preprint arXiv:1912.00818, 2019.

[6] A. Krizhevsky, G. Hinton, and others, "Learning multiple layers of features from tiny images," 2009.

[7] B. Liu, Y. Guo, and X. Chen, "PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization," in Proceedings of the Web Conference 2021, 2021, pp. 923–934.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[9] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," arXiv preprint arXiv:1706.01307, 2017.

[10] P. E. Danielsson, "Euclidean distance mapping," Computer Graphics and image processing, vol. 14, no. 3, pp. 227–248, 1980.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[12] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[13] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in neural information processing systems, vol. 32, pp. 8026–8037, 2019.

[14] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," A Practical Guide, 1st Ed., Cham: Springer International Publishing, vol. 10, p. 3152676, 2017.

[15] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in Annual Cryptology Conference, 2012, pp. 643–662.

[16] M. Abadi et al., "Deep learning with differential privacy," in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.

[17] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," arXiv preprint arXiv:2002.10619, 2020.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in International Conference on Machine Learning, 2017, pp. 1126–1135.

Improving Mutation-Based Fault Localization via Mutant Categorization

Xia Li
Department of Software Engineering and Game Design, Kennesaw State University
xli37@kennesaw.edu

Durga Nagarjuna Tadikonda
Department of Software Engineering and Game Design, Kennesaw State University
dtadikon@students.kennesaw.edu

***Abstract*—Fault localization is one of the most important activities in software debugging. Among various fault localization techniques, mutation-based fault localization (MBFL) has been commonly studied with its promising performance. However, MBFL should be improved further by incorporating more useful program information. In this paper, we propose MuCatFL, a novel and lightweight technique for better MBFL via mutant categorization. In details, after executing the original test suite against all generated mutants, we categorize the mutants into two groups, positive mutants and negative mutants, to rank the tied program elements. We evaluate MuCatFL by performing an extensive study on 395 real software faults from the widely used benchmark Defects4J. The experimental results show that MuCatFL can significantly outperform MBFL techniques (e.g., localizing 138 faults within the Top-1 position on method-level, 43.75% more than traditional Metallaxis technique). We also investigate that only positive mutants can contribute to the effectiveness of MBFL. Our findings can also provide guidance for the strategies to reduce the execution cost of MBFL.**

***Index Terms*—Software debugging, Fault localization, Mutation testing**

I. Introduction

In modern software development, bugs (a.k.a., faults) are prevalent and inevitable due to the complexity of software systems. They have been widely recognized as notoriously costly and disastrous. For example, Tricentis.com [1] investigated and reported software bugs impacting 3.7 billion users and $1.7 trillion in assets. The first step of debugging is to localize the potential faulty location(s). However, manual fault localization can be time-consuming and error-prone due to the huge code volume. To solve this problem, researchers have proposed various automated fault localization (FL) techniques [2], [3], [4], [5] to help reduce manual efforts. The basic idea of fault localization techniques is to generate a ranked list of program elements (e.g., methods or statements) according to the descending order of their suspiciousness values. Developers can use the ranked list to manually check each element to find and fix the faults in the faulty program. Thus, the target of FL techniques is to rank the faulty elements as high as possible in the ranked list.

In the literature, *spectrum-based fault localization* (SBFL) [6], [2], [7] has been intensively studied since it simply considers the coverage information of failed/passed tests which can be easily collected by many coverage analysis tools. The basic intuition of SBFL is that one program element is more suspicious if it is executed/covered by more failed tests than passed tests. Based on the intuition, various SBFL techniques are proposed (such as Tarantula [2], Ochiai [3], DStar [8], Jaccard [6]) to utilize statistical analysis to compute the suspiciousness values of program elements. Despite the lightweightness, SBFL still has some limitations. For example, some faulty elements may also be covered by passed test cases coincidentally and failed test cases may still cover non-faulty elements. To overcome this limitation, researchers propose *mutation-based fault localization* (MBFL) [5], [4] to consider the actual impact information (a.k.a., killing information) of each program element on the test outcomes by mutating program source code. To localize faulty elements more precisely, typical MBFL techniques such as Metallaxis [4] and MUSE [5] generate mutants for the original program by changing statements syntactically based on predefined rules (also called *mutation operators*, such as changing `a+b` into `a-b`). All the generated mutants are then executed by original test suite and the new execution results of test cases are used for more precise fault localization.

MBFL has been proved to be more effective than SBFL in real bugs [9], but its accuracy is still not promising as expected for many faults. The potential reason is that impact information alone cannot help distinguish some tied program elements (i.e., many elements share same suspiciousness values with the actual buggy elements). Various studies are proposed to improve the accuracy of MBFL. For example, MuSim [10] is presented by identifying the faulty statements based on test case proximity to different mutants. However, these techniques heavily rely on complex computation or analysis (e.g., using test case proximity or neural network). In this paper, therefore, we propose MuCatFL, a novel and lightweight technique for better MBFL by mutant categorization. In details, after executing the original test suite against all generated mutants, we categorize the mutants into two groups, positive mutants and negative mutants, to rank the tied program elements. To evaluate MuCatFL, we perform an extensive study on 395 real software faults from the widely used benchmark Defects4J (V1.2.0) [11]. The experimental results show that MuCatFL can significantly outperform MBFL techniques (e.g.,

DOI:10.18293/SEKE2022-157

localizing 138 faults within the Top-1 position, 43.75% more than traditional Metallaxis technique). This paper makes the following contributions:

- **Technique** A novel and lightweight MBFL technique MuCatFL via mutant categorization.
- **Study** An extensive study on localizing real faults to demonstrate the effectiveness of MuCatFL.

The structure of this paper is as follows. In Section II, we introduce the basis of fault localization and related studies on MBFL. In Section III, we propose the framework and algorithm of MuCatFL. Next, we introduce the experimental study design and analyze the experimental results in Section IV and Section V. Finally, we conclude our paper in Section VI and Section VII.

II. Background and Related Work

A. Fault Localization

Spectrum-Based Fault Localization. The basic idea of SBFL is that program elements covered by more failed test cases tend to be more suspicious. SBFL takes the coverage information between program elements and test suite as input and output a suspicious ranked list in the descending order of suspiciousness values. To date, various SBFL techniques have been proposed such as Tarantula [12], SBI [7], Ochiai [3], Jaccard [6], etc. Even though these techniques use different statistical analysis, they mainly rely on the following components for calculation: (1) the number of all failed/passed tests, i.e., n_f/n_p, (2) the number of failed/passed tests executing program element e, i.e., $n_f(e)$/$n_p(e)$, and (3) the number of failed/passed tests that do not execute program element e, i.e., $n_f(\bar{e})$/$n_p(\bar{e})$. For example, SBI formula can calculate the suspiciousness value of element e as $Susp(e) = \frac{n_f(e)}{n_f(e)+n_p(e)}$. All program elements are ranked based on their suspiciousness values calculated from the formulae and the ranked list is provided to developers to check and repair bugs manually.

Mutation-Based Fault Localization. SBFL has one major limitation. In buggy programs, failed tests may cover non-buggy program elements that do not contribute to the program failure and buggy program elements are also executed by passed tests. MBFL techniques overcome this limitation by considering more effective impact information between test suite and program elements. The first MBFL technique Metallaxis [13], [4] is proposed based on the following intuition: if one mutant impacts failed tests (e.g., the test outcomes change after mutation), its corresponding program element may have caused the failures so that this element should have higher probability to be faulty than others. In Metallaxis, mutants that have impacts on tests are viewed as program elements covered by the tests while the others as uncovered. By simulating the coverage information, Metallaxis applies traditional SBFL formulae to calculate each mutant's suspiciousness value. Finally, the maximum value of mutants is treated as the suspiciousness value of corresponding program element. For example, based on the SBI formula, the suspiciousness value of mutant m can be calculated as $Susp(m) = \frac{n_f^{(m)}(e)}{n_f^{(m)}(e)+n_p^{(m)}(e)}$, where $n_f^{(m)}(e)$/$n_p^{(m)}(e)$ is the number of failed/passed tests whose outcomes are changed due to the mutant m on element e. Another popular MBFL technique is called MUSE [5] which shares similar intuitions with Metallaxis that mutating faulty program elements may cause more failed tests to pass than mutating correct elements and mutating correct elements may cause more passed tests to fail than mutating faulty elements. Besides the two basic MBFL techniques, TraPT [9] extends them to obtain more detailed impact information by transforming test outcomes to extract various test failure messages for better fault localization. For example, TraPT considers MBFL results with the following four different types of test outcomes: (1) Type1: pass/fail information, (2) Type2: exception type, (3) Type3: exception type and exception message, and (4) Type4: exception type, message, and the full stack trace. The four types of test outcomes will generate different impact information. Please note that Metallaxis uses Type4 test failure outcomes and MUSE uses Type1 test failure outcomes according to TraPT.

B. Improvement of Mutation-Based Fault Localization

Previous studies [4], [5], [13] have demonstrated the effectiveness of basic MBFL. However, MBFL still has two major limitations regarding its efficiency and accuracy. The first limitation is that MBFL suffers from the extremely high mutant execution cost problem since it generates a significant number of mutants for the program under test, and each mutant must be executed by all test cases [14], [15]. To overcome this limitation, various studies have been proposed. FTMES [16] is proposed to use only failed tests to execute against mutants and avoid the execution of passed test cases by replacing the impact information with coverage information. IETCR [15] is introduced to reduce the execution of test cases by calculating the entropy change of tests and selecting a proportion of them according to the entropy values. SMBFL [17] is proposed to reduce the execution cost by examining only the statements in the dynamic slice of the program under test to reduce the number of statements to be mutated. Another limitation of MBFL is that impact information alone cannot distinguish many tied program elements so that more advanced program features should be further extracted. For example, MuSim [10] is presented by identifying the faulty statements based on test case proximity to different mutants, 33.21% more effective than existing fault localization techniques such as DStar, Tarantula and Ochiai. In this paper, we propose a novel and lightweight approach to improve the accuracy of MBFL, but our results and findings can provide valuable guidance for more efficient MBFL.

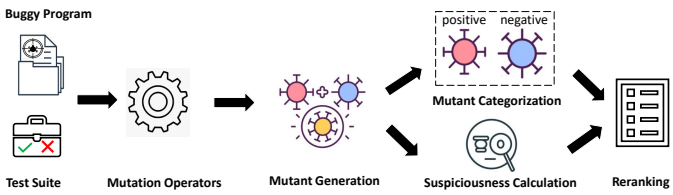


Fig. 1: MuCatFL framework

III. Approach

In this section, we introduce the framework (Section III-A) and algorithm (Section III-B) of our new technique MuCatFL. We also present a real-world example in Section III-C.

Algorithm 1: MuCatFL Algorithm

Input: Faulty program P, test suite T, coverage information C, SBFL formula F, a group of mutators Op
Output: A ranked list S

1 $P' \leftarrow$ A set of elements that covered by failed tests based on the coverage C
2 **for** *element* $e \in P'$ **do**
3 $M(e) \leftarrow$ a set of mutants for e based on Op
4 $N_{positive}(e) \leftarrow 0$ which is the number of positive mutants for e
5 $N_{negative}(e) \leftarrow 0$ which is the number of negative mutants for e
6 **for** $m \in M(e)$ **do**
7 $Sus(m) \leftarrow$ $F(n_p^{(m)}(e), n_f^{(m)}(e), n_f^{(m)}(\bar{e}), n_p^{(m)}(\bar{e}))$
8 **if** $n_f^{(m)}(e) > 0$ **then**
9 $N_{positive}(e)$ ++
10 **else**
11 **if** $n_p^{(m)}(e) > 0$ **then**
12 $N_{negative}(e)$++
13 **end**
14 **end**
15 **end**
16 $Sus(e) \leftarrow$ Max($Sus(m)$)
17 **end**
18 List $S' \leftarrow$ ranked elements according to initial suspiciousness values (descending order)
19 List $S'' \leftarrow$ ranked elements from S' according to the number of positive mutants (descending order)
20 List $S''' \leftarrow$ ranked elements from S'' according to the number of negative mutants (ascending order)
21 Final ranked list S $\leftarrow S'''$

A. Framework of MuCatFL

Figure 1 shows the framework with detailed procedures of MuCatFL. The first several steps are same with traditional MBFL. Given a buggy program under test and its test suite, traditional MBFL techniques apply mutation testing to generate a huge number of mutants for each program statement based on predefined mutation operators (also called mutators). Next, the test suite including all failed and passed test cases is executed against all mutants to record the impact information. The preliminary suspiciousness values can be calculated based on various formulae to get the initial ranked list of the program elements.

In many cases, even the impact information is adopted, some program elements still are tied with the same suspiciousness value. To overcome this limitation, in MuCatFL, we also collect various types of mutants for each program element based on the impact information. We define the following mutant categories based on Type1 failure message (pass/fail information): (1) **positive mutant** that makes **any** failed test pass, (2) **neutral mutant** that makes all test cases unchanged and (3) **negative mutant** that makes **all** failed tests still fail and makes any passed test fail. Since neutral mutants do not contribute to the impact information (no change of test outcomes), we only consider positive mutants and negative mutants in the detailed implementations. Please note that the category definitions based on Type4 failure message is different due to the different impact information (e.g., Type1 for MUSE and Type4 for Metallaxis). For example, the positive mutant according to Type4 failure message represents the mutant that makes the exception type, message and full stack trace of **any** failed test change (even the failed test still fails on the mutant). According to the categories of mutants, we further rank the tied program elements to break the ties based on the following intuitions: (1) if a program element can generate more positive mutants, it has higher probability to be faulty, and (2) if a program element can generate more negative mutants, it has lower probability to be faulty. To this end, for the tied program elements with same suspiciousness value computed by traditional MBFL, we rank the element **higher** if it generates more positive mutants than others. If there are still tied program elements, we further rank the element **lower** if it generates more negative mutants than others. Finally, we can get the final ranked list for all program elements.

B. Algorithm of MuCatFL

Algorithm 1 describes more implementation details of MuCatFL. The entries of MuCatFL include faulty program P, test suite T (with passed tests and failed tests), coverage information C between P and T. It also includes SBFL formula F used by MBFL techniques and a group of predefined mutation operators Op. Next, we explain the details of the algorithm. As the previous study TraPT [9], only suspicious mutants occurring on program elements executed by failed tests contribute to MBFL, so we collect a set of elements P' covered by failed tests in Line 1. From Line 2 to Line 17, we iterate all elements from P' to collect required components for MuCatFL. In Line 3, we generate a set of mutants for each element e based on a group of predefined mutation operators Op. In Line 4 and Line 5, we initialize two variables $N_{positive}(e)$ and $N_{negative}(e)$ to store the numbers of positive mutants and negative mutants for each element. From Line 6 to Line 15, we iterate all mutants for e to calculate their suspiciousness values based on the basic MBFL. We also check if the mutant is positive or negative. In detail, we first check that if $n_f^{(m)}(e)$ is greater than 0 then we add 1 to $N_{positive}(e)$. Otherwise, if $n_p^{(m)}(e)$ is greater than 0, we add 1 to $N_{negative}(e)$. In Line 16, we assign the maximum suspiciousness value of all mutants of e as its final value. Finally, we rank all elements based on the orders of suspiciousness values, the number of positive and negative mutants to get the final ranked list (Line 18 to Line 21).

```
public LegendItemCollection getLegendItems() {
     ...
    int index = this.plot.getIndexOf(this);
    CategoryDataset dataset = this.plot.getDataset(
        index);
--- if (dataset != null) {
+++ if (dataset == null) {
        return result;
    }
    int seriesCount = dataset.getRowCount();
    ...
}
```

Fig. 2: Example of buggy and fixed statements from Chart-1

```
public DefaultDrawingSupplier(Paint[] paintSequence,
     Paint[] fillPaintSequence,...) {

        ...
        this.fillPaintSequence = fillPaintSequence;
        ...
        ...
    }
```

Fig. 3: Example of non-buggy statement from Chart-1

C. Example of MuCatFL

In this section, we use a real-world example from Defects4J (V1.2.0) [11], a widely used Java bug benchmark in the field of software testing and debugging, to demonstrate the effectiveness of our technique MuCatFL. We use Chart-1 which denotes the first buggy version from JFreeChart [18] project. The buggy statement (i.e., `if (dataset != null)` is located in the method `getLegendItems()` of Class `AbstractCategoryItemRenderer` as shown in Figure 2. Based on the traditional MBFL, this buggy statement shares the same suspiciousness value with one non-buggy statement in Class `DefaultDrawingSupplier` as shown in Figure 3. By means of mutant categorization, we observe that there are two positive mutants generated on this buggy statement that can make the failed test pass according to the mutators `RemoveConditionalMutator` and `NegateConditionalsMutator` from the widely used mutation testing tool PIT [19] while there is only one positive mutant generated by the mutator `MemberVariableMutator` for the non-buggy statement. This example demonstrates the effectiveness of MuCatFL to differentiate the buggy elements and non-buggy elements via mutant categorization, indicating that program elements with more positive mutants are prone to be faulty.

IV. Study Design

In this work, we aim to investigate the following research questions:

- **RQ1:** How does MuCatFL perform in localizing real faults compared with traditional MBFL techniques?
- **RQ2:** How do positive mutants or negative mutants impact the performance of MuCatFL separately?
- **RQ3:** What mutators can generate most positive mutants for the studied real-world faults?

A. Implementation and Tool Supports

In this paper, we perform on-the-fly bytecode instrumentation using ASM [20] and Java Agent [21] to collect the required coverage information. We apply the widely used mutation testing framework PIT (Version 1.1.5) [19] to perform mutation testing. Following the previous study TraPT [9], we use all 16 mutation operators available in PIT-1.1.5 and modify PIT to collect the required impact information. For example, we modify PIT to enable it executing on programs with failed tests and force it to execute each mutant against the remaining tests even the mutant is killed by earlier tests. We implement MUSE and 5 widely used traditional SBFL formulae (Tarantula [2], Ochiai [3], DStar [8], Jaccard [6] and SBI [7]) for Metallaxis. We use 395 faulty versions from all the 6 projects (Lang, Chart, Time, Math, Mockito and Closure) in widely used Defects4J benchmark (V1.2.0) [11].

B. Evaluation Metrics

Many prior studies [22], [23], [24], [25] perform fault localization techniques on method-level, i.e., localizing faulty methods among all source code methods, since statement-level fault localization may be too fine-grained without context information [26] and class-level fault localization is too coarse-grained [27]. In these studies, the suspiciousness value of one method is assigned as the the maximum suspiciousness value of all mutants generated in this method. In this paper, we also evaluate MuCatFL on method-level inspired by other studies but make some changes since the number of mutants is involved. In detail, we firstly rank all statements in each source code method based on Algorithm 1. Next, we find the top-rank statement and assign its suspiciousness value, the number of positive mutants and the number of negative mutants to its corresponding method. Finally, we rank all source code methods based on the Line 18-20 in Algorithm 1. We use following evaluation metrics to evaluate various MBFL techniques. (1) Top-N (Top-1, Top-3, and Top-5 in our study) metric simply represents the exact position of the buggy elements in the ranked list. The motivation to use Top-N metric is that most developers will stop using debugging tools if they cannot return the actual buggy elements within the Top-5 positions [27]. (2) MFR (mean first rank). For a buggy version with multiple buggy elements, we use MFR to compute the mean of the first buggy element's rank for each buggy version since the localization of the first buggy element can be a guide to the rest of buggy elements. (3) MAR (mean average rank) is simply the mean of the average of all buggy elements' ranks for each buggy version.

V. Result Analysis

A. RQ1 - Performance of MuCatFL

In this section, we investigate the effectiveness of MuCatFL compared with traditional MBFL techniques (Metallaxis and MUSE). Figure 4 shows the overall fault localization results on all studied subjects (i.e., Lang, Chart, Time, Math, Mockito and Closure from the Defects4J benchmark) in terms

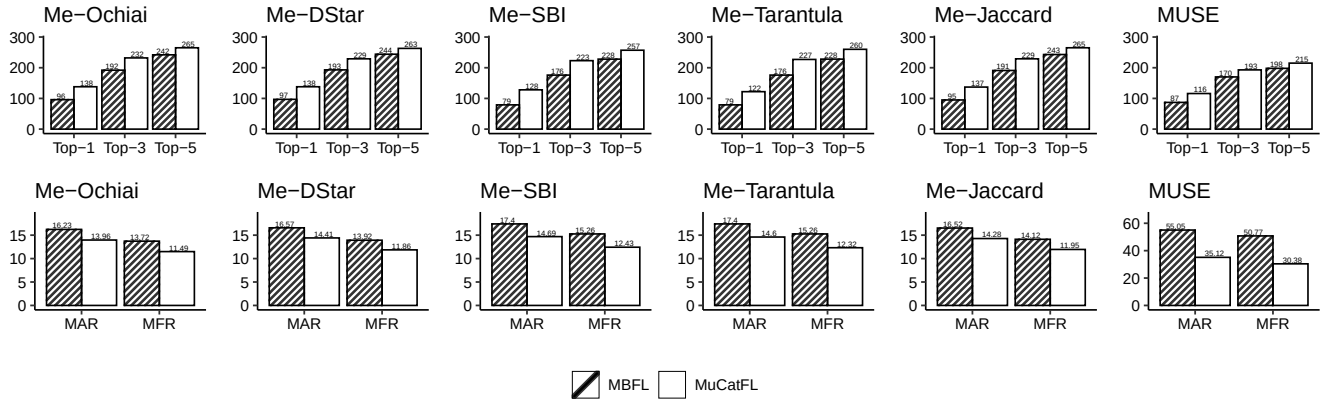


Fig. 4: Results of MuCatFL compared with Metallaxis and MUSE

TABLE I: Impacts of positive and negative mutants

Tech Name	Top-1	Top-3	Top-5	MFR	MAR
Me-Ochiai	96	192	242	13.72	16.23
MuCatFL(P and N)	138	232	265	11.49	13.96
MuCatFL(Only P)	133	230	263	12.08	14.71
MuCatFL(Only N)	104	205	246	12.93	15.28

of Top-1, Top-3, Top-5, MFR and MAR. The upper sub-figures represent the Top-N results and bottom sub-figures indicate the MFR/MAR results. Each pair of bars in the sub-figures represents the comparison between MuCatFL and one traditional MBFL with different formulae. Please note that in the figure we use "Me-formula" to represent Metallaxis with corresponding SBFL formula. In these figures, higher Top-N value and lower MFR/MAR value indicate a better localization technique. From the figures, we have following observations. First, MuCatFL with mutant categorization outperforms traditional MBFL techniques for **all** SBFL formulae. For example, in total, Metallaxis with Ochiai formula can localize 96 faulty methods within Top-1, while MuCatFL is able to localize 138 faulty methods, 43.75% more effective than traditional MBFL technique. Furthermore, in terms of MAR, MuCatFL for Ochiai is 13.96, 13.99% more precise than Metallaxis with Ochiai (16.23). Second, in terms of MAR/MFR, MuCatFL can improve Metallaxis by less than 20% for the five formulae. However, MUSE can be improved by 36.2% and 40.16% when we consider different categories of mutants. The potential reason can be that MUSE only considers pass/fail information so that there are more rooms to be improved by utilizing positive mutants and negative mutants.

B. RQ2 - Impacts of Positive Mutants or Negative Mutants

In the RQ1, we compare MuCatFL with traditional MBFL techniques by considering both positive and negative mutants. However, whether both of them contributes to MuCatFL has not been studied. In this section, we investigate the effectiveness of MuCatFL by considering positive or negative mutants **separately**. Table I shows the fault localization results with only Ochiai formula for different configurations since MuCatFL with Ochiai can achieve the best performance according to RQ1. In this table, Me-Ochiai represents traditional MBFL technique and MuCatFL (P and N) indicates MuCatFL with both positive and negative mutants. Also, MuCatFL (Only P) represents MuCatFL only considering positive mutants while MuCatFL (Only N) represents MuCatFL only considering negative mutants. From the table, we have following observations. First, both positive and negative mutants are helpful for MuCatFL. For example, in terms of Top-1, MuCatFL (P and N) can localize 138 faulty methods within Top-1, more than any other configurations. Second, only positive mutants can still contribute to promising performance of MuCatFL. In detail, MuCatFL (Only P) can help localize 133 faulty methods within Top-1, very close to MuCatFL (P and N). However, MuCatFL with only negative mutants performs worse than that with only positive mutant (localizing 104 bugs within Top-1). Such findings demonstrate that failed tests should be more important than passed tests when localizing faults for MBFL, indicating the potential improvement of accuracy for some cost reduction strategies (e.g., FTMES [16] with the execution of only failed tests against all mutants).

C. RQ3 - Impacts of Mutation Operators for Positive Mutants

In MBFL, we generate mutants by applying different mutation operators, and the findings in RQ2 show that only positive mutants can contribute to MuCatFL. In this section, we investigate what mutation operators can mostly lead to positive mutants in terms of both Type1 and Type4 test failure messages. We count the number of positive mutants with their corresponding mutators in Table II and Table III, accordingly. Please note that we only include 6 most frequent mutators in the tables, which can reveal some interesting findings. In the two tables, the first column represents the mutators from PIT and the second column indicates the number of positive mutants generated by the corresponding mutators. From the two tables, we can find that the top 6 mutators are exactly same for Type1 and Type4 failure message even the orders are different, indicating that program elements that can be mutated by these mutators tend to be faulty. This finding can be also applied to reduce the huge execution cost of MBFL. For example, mutants generated by top frequent mutators can have higher execution priorities, or only the top frequent mutators can be selected to generate mutants.

TABLE II: Mutators generating most positive mutants in terms of Type1 failure message

Mutator	# of positive mutants
NonVoidMethodCallMutator	4686
NegateConditionalsMutator	4170
RemoveConditionalMutator_EQUAL_ELSE	2698
InlineConstantMutator	1998
ReturnValsMutator	1960
RemoveConditionalMutator_EQUAL_IF	1912

TABLE III: Mutators generating most positive mutants in terms of Type4 failure message

Mutator	# of positive mutants
NonVoidMethodCallMutator	115912
NegateConditionalsMutator	66043
ReturnValsMutator	47439
InlineConstantMutator	47218
RemoveConditionalMutator_EQUAL_IF	45171
RemoveConditionalMutator_EQUAL_ELSE	31347

VI. Threats to Validity

The main threat to *internal* validity is from our implementation. To reduce this threat, we implement our techniques by utilizing state-of-the-art tools and frameworks, such as ASM and PIT. The main threat to *external* validity mainly lies in the selection of the studied subjects. To reduce this threat, we evaluate on more real-world projects. The main threat to *construct* validity is that the measurements used may not fully reflect real-world situations. To reduce this threat, we use Top-N, MAR and MFR metrics, which have been widely used in previous studies [9], [25], [28], [24].

VII. Conclusion

In this paper, we propose MuCatFL, a novel and lightweight technique for better MBFL via mutant categorization. In details, after executing the original test suite against all generated mutants, we categorize the mutants into two groups, positive mutants and negative mutants, to rank the tied program elements. We evaluate MuCatFL by performing an extensive study on 395 real software faults from the widely used benchmark Defects4J. The experimental results show that MuCatFL can significantly outperform MBFL techniques (e.g., localizing 138 faults within the Top-1 position, 43.75% more than traditional Metallaxis technique).

References

[1] "Tricentis reports," 2018. [Online]. Available: https://www.tricentis.com/blog/how-to-avoid-the-tricentis-software-fail-watch/

[2] J. A. Jones and M. J. Harrold, "Empirical evaluation of the tarantula automatic fault-localization technique," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, 2005, pp. 273–282.

[3] R. Abreu, P. Zoeteweij, and A. J. Van Gemund, "An evaluation of similarity coefficients for software fault localization," in *2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*. IEEE, 2006, pp. 39–46.

[4] M. Papadakis and Y. Le Traon, "Metallaxis-fl: mutation-based fault localization," *Software Testing, Verification and Reliability*, vol. 25, no. 5-7, pp. 605–628, 2015.

[5] S. Moon, Y. Kim, M. Kim, and S. Yoo, "Ask the mutants: Mutating faulty programs for fault localization," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*. IEEE, 2014, pp. 153–162.

[6] R. Abreu, P. Zoeteweij, and A. J. Van Gemund, "On the accuracy of spectrum-based fault localization," in *Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION (TAICPART-MUTATION 2007)*. IEEE, 2007, pp. 89–98.

[7] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan, "Scalable statistical bug isolation," *ACM Sigplan Notices*, vol. 40, no. 6, pp. 15–26, 2005.

[8] W. E. Wong, V. Debroy, Y. Li, and R. Gao, "Software fault localization using dstar (d*)," in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*. IEEE, 2012, pp. 21–30.

[9] X. Li and L. Zhang, "Transforming programs and tests in tandem for fault localization," *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3133916

[10] A. Dutta, A. Jha, and R. Mall, "Musim: Mutation-based fault localization using test case proximity," *International Journal of Software Engineering and Knowledge Engineering*, vol. 31, no. 05, pp. 725–744, 2021.

[11] R. Just, D. Jalali, and M. D. Ernst, "Defects4j: A database of existing faults to enable controlled testing studies for java programs," in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, 2014, pp. 437–440.

[12] J. A. Jones, M. J. Harrold, and J. T. Stasko, "Visualization for fault localization," in *in Proceedings of ICSE 2001 Workshop on Software Visualization*, 2001.

[13] M. Papadakis and Y. Le Traon, "Using mutants to locate" unknown" faults," in *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. IEEE, 2012, pp. 691–700.

[14] M. Kooli, F. Kaddachi, G. Di Natale, A. Bosio, P. Benoit, and L. Torres, "Computing reliability: On the differences between software testing and software fault injection techniques," *Microprocessors and Microsystems*, vol. 50, pp. 102–112, 2017.

[15] H. Wang, B. Du, J. He, Y. Liu, and X. Chen, "Ietcr: An information entropy based test case reduction strategy for mutation-based fault localization," *IEEE Access*, vol. 8, pp. 124 297–124 310, 2020.

[16] A. A. L. de Oliveira, C. G. Camilo-Junior, E. N. de Andrade Freitas, and A. M. R. Vincenzi, "Ftmes: A failed-test-oriented mutant execution strategy for mutation-based fault localization," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 155–165.

[17] N. Bayati Chaleshtari and S. Parsa, "Smbfl: slice-based cost reduction of mutation-based fault localization," *Empirical Software Engineering*, vol. 25, no. 5, pp. 4282–4314, 2020.

[18] "Jfreechart website," 2022. [Online]. Available: https://github.com/jfree/jfreechart

[19] "Pit mutation testing system," 2022. [Online]. Available: http://pitest.org/

[20] "Asm java bytecode manipulation and analysis framework," 2022. [Online]. Available: https://asm.ow2.io/

[21] "Java programming language agents," 2022. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/lang/instrument/package-summary.html

[22] T. Dao, L. Zhang, and N. Meng, "How does execution information help with information-retrieval based bug localization?" in *Proceedings of the 25th International Conference on Program Comprehension*, 2017, pp. 241–250.

[23] X. Li, W. Li, Y. Zhang, and L. Zhang, "Deepfl: Integrating multiple fault diagnosis dimensions for deep fault localization," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 169–180.

[24] M. Zhang, X. Li, L. Zhang, and S. Khurshid, "Boosting spectrum-based fault localization using pagerank," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2017, pp. 261–272.

[25] J. Sohn and S. Yoo, "Fluccs: using code and change metrics to improve fault localization," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, 2017, pp. 273–283.

[26] C. Parnin and A. Orso, "Are automated debugging techniques actually helping programmers?" in *Proceedings of the 2011 international symposium on software testing and analysis*, 2011, pp. 199–209.

[27] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 165–176.

[28] J. Xuan and M. Monperrus, "Learning to combine multiple ranking metrics for fault localization," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 191–200.

WBS: Weighted Backtracking Strategy for Symbolic Testing of Embedded Software

Varsha P Suresh[1], Sujit Kumar Chakrabarti[1], Athul Suresh[1], and Raoul Jetley[2]

[1]International Institute of Information Technology Bangalore, Bengaluru, India
[2]ABB Corporate Research, Bengaluru, India

Abstract

Symbolic execution is an important program analysis technique that has found a number of applications in the last fifteen years or so. Popular symbolic execution approaches use backtracking when faced with infeasibility along a path being explored. A simple backtracking strategy (i.e. backtracking by a single decision node) may suffice when the goal is to cover the entire control flow graph (CFG). However, if the goal is to cover specific parts of the CFG through a single path, simple backtracking may lead to non-optimality or even non-termination. In this paper, we present weighted backtracking strategy (WBS) that exploits previous knowledge about the program behaviour to compute 'good' candidates as destinations of backtracking. We have integrated our heuristic to SymTest, a symbolic testing framework for embedded systems. Experiments with case-studies have demonstrated that WBS improves SymTest's performance both in its ability to achieve termination as well as in computing shorter test sequences compared to the original approach. SymTest with WBS generates shorter test sequences compared to several other existing test generation approaches based on symbolic execution.

1 Introduction

Large amount of time and effort of software development is spend in testing of the software. In automated test case generation the software under test is analysed and then test data are generated. The generated test data is used for test execution. Test execution can be simulation of the software or target testing, where the software is executed in the deployment environment. In case of real time embedded software the cost of testing goes up dramatically because such reactive systems requires interaction with other physical subsystems, human interfaces etc. Thus cost of testing has a direct impact on cost of software quality, which necessitates the optimization of software testing. In embedded systems and systems involving interactions with other physical subsystems, the *test sequence length* (defined in Section 2) plays a vital role in the cost of testing. As the test sequence length increases the interaction between physical subsystems increases which in turn increases the cost of testing. Even though there has been significant advancement in symbolic execution in recent years, there has not been much effort put towards obtaining short test sequences.

SymTest[1] is a symbolic execution framework used to generate short test sequences for embedded software. The main idea in SymTest is to explore only one path that covers the target edges. If the first path computed is feasible, SymTest indeed provides very good results in terms of the length of the test sequence (path) generated. However, if the path chosen is infeasible, the guarantee of the shortness of the generated test sequence is severely curtailed. Backtracking by one decision is the backtracking approach of SymTest which provides no assurance that the alternative path thus explored even terminates. We present a heuristic, *Weighted Backtracking Strategy* (WBS) which computes a candidate backtracking destination which may be more than one decision edge behind the last decision edge. This approach is based on weight calculations obtained from earlier runs of the system. We have implemented WBS as a part of SymTest and tested it against a number of case studies which reflects the path exploration problem. WBS aided in generating test sequences in multiple cases where SymTest fails to terminate, and generates shorter test sequences compared to SymTest and several other existing testing approaches based on symbolic execution.

The rest of this paper is organized as follows: A motivating example and an overview of the related works is discussed in Section 2. Section 3 explains the WBS and the integration of WBS in SymTest. Section 4 presents the experiment details and the obtained results. Section 5 concludes the paper.

2 Motivation and Related Works

In structural testing, the goal is to execute the system so as to cover certain parts of the system. In first-time testing, we may want to cover the entire system. But in regression

DOI reference number: 10.18293/SEKE2022-147

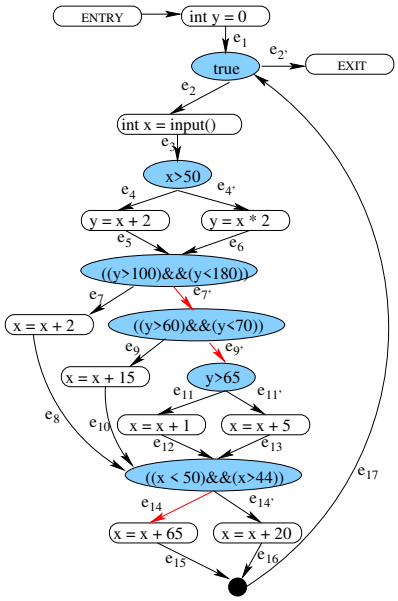


Figure 1: CFG

testing, depending on the requirement, certain parts of the system may need to be covered. In our work, we consider *edge coverage* as the coverage criterion. Our testing aims to cover a set T of edges, called as *target edges* which may be the set of all edges in the control flow graph of the system, or a proper subset thereof. A *test sequence* is a sequence of unit interactions between the system and the environment that completely covers a given target edge set T. During test execution, often the most time consuming (or resource intensive) process is the interaction with the environment which may involve network delays, mechanical movements and human actions. Hence, test sequence length, measured as the number of unit interactions of the system with the environment, may have a direct implication on cost of testing.

SymTest ensures generating shorter test sequences if the initial path obtained from the control flow graph is satisfiable. If backtracking occurs, then due to path explosion problem the shorter test sequences are not guaranteed. Consider the control flow graph (CFG) shown in Fig. 1. The edges marked in red in the CFG are the target edges we want to cover during testing. The path, $e_1e_2e_3e_4e_5e_{7'}e_{9'}e_{11}e_{12}e_{14}e_{15}e_{17}e_{2'}$ is computed and on performing symbolic execution on the obtained path, it is found to be unsatisfiable. Next step is to figure out the alternative optimal feasible path which covers the target edge. Optimal path here means syntactically shortest path that covers the target edges. Search strategy in SymTest follows backtracking by one decision edge to find the next path. In the example given, the new path obtained is also found to be unsatisfiable by the SMT solver. Backtracking proceeds further to search the feasible path and SymTest fails to terminate as it repeatedly takes the edge e_4 on each iteration in its search. Therefore this strategy does not ensure finding optimal path after backtracking. Contrary to this when we use WBS in SymTest it redirects the execution along the edge of $e_{4'}$, the resulting path is : $e_1e_2e_3e_{4'}e_6e_{7'}e_{9'}e_{11}e_{12}e_{14}e_{15}e_{17}e_{2'}$. This turns out to be feasible, thus resulting in a successful generation of test sequence of short length.

Classical search strategies - depth first search (DFS) have been used in tools DART [2], CUTE [3]. JPF [4] [5] has the capability to choose the search strategy DFS or BFS. JPF also comprises structural heuristics for path exploration. CREST and KLEE are two concolic testing tools which have been adopted for testing in industrial applications [6][7]. CREST uses control flow directed search, where static structure of the program is considered to explore program's path space. The other search strategies devised for concolic testing in CREST includes bounded depth-first search, uniform random search, random branch search [8]. The commonly used search strategies in KLEE[9] are - Coverage-Optimized Search, Depth First Search, Random Path Select, Random State Search. Fitnex [10] is a search strategy used to guide path exploration in dynamic symbolic execution to achieve test target coverage. Program-derived fitness functions were used to calculate the fitness values for the explored paths. The fitness function measures how close an explored path is in achieving test target coverage. For effective exploration, the core Fitnex strategy has been integrated with other search strategies.

Search strategies used in the symbolic execution tools like KLEE, CREST focus on achieving high code coverage. They do not ensure target edge coverage in least number of iterations. On the other hand the integration of WBS in original SymTest focuses on target edge coverage in fewest number of iterations.

3 Proposed Approach

Before explaining the WBS in detail in Section 3.2, notations used in the paper are introduced here and Section 3.1 illustrates how WBS is used in SymTest to improve the generation of test sequences.

Notations Some of the notational conventions followed in the sequel is given here.

- $\mathcal{P}$: Program under test
- $\mathcal{G}$: Control flow graph of $\mathcal{P}$
- $V(\mathcal{G})$: Node set of $\mathcal{G}$
- $E(\mathcal{G})$: Edge set of $\mathcal{G}$
- Nodes in $V(\mathcal{G})$: represented by names like n, n', n_i, , where $i \in \mathbb{N}$
- Edges in $E(\mathcal{G})$: represented by names like e, e', e_i, where $i \in \mathbb{N}$
- Each decision node in $\mathcal{G}$ has two outgoing decision edges e and e'. We say that $e = flip(e')$ and $e' = flip(e)$.
- T: Target edge set
- $\mathcal{T}$: Computational tree
- $V(\mathcal{T})$: Node set of $\mathcal{T}$

- $E(\mathcal{T})$: Edge set of $\mathcal{T}$
- Nodes in $V(\mathcal{T})$ are represented by names like $\mathbf{n}$, $\mathbf{n}'$, $\mathbf{n_1}$, $\mathbf{n_i}$, where $i \in \mathbb{N}$
- $\rightarrow_{\mathcal{G}}$ is a relation between two decision edges $e_i, e_j \in E(\mathcal{G})$ such that e_i is an immediately preceding decision edge to e_j in at least one run of the program.

3.1 SymTest-WBS

A revised version of the SymTest algorithm using WBS is presented in Algorithm 1. We use FINDCFPATH [1] to find an optimal syntactic path through the control flow graph that covers all members in the target edge set T. The path thus computed is pushed into the $stack$ as a sequence of decision edges. We symbolically execute along this path

Algorithm 1 SymTest-WBS

```
1:  procedure SYMTEST(G, T, W)
2:      stack ← <>
3:      while true do
4:          stack ← FINDCFPATH(G, T, stack)
5:          sympath ← SYMEX(G, stack)
6:          pc ← PC(sympath)
7:          tf, M ← SOLVE(pc)
8:          if tf then
9:              return M
10:         else
11:             b ← BTP(G, stack, W, T)
12:             stack ← BACKTRACK(stack, flip(b))
13:             PUSH(stack, b)
```

giving us the symbolic trace along that path ($sympath$). We convert this into a logic formula and input it to an SMT solver (SOLVE). If the formula is found satisfiable (indicated by true value of the true or false (tf) component of the value returned by SOLVE), our search was successful; the test input can be directly extracted from the model M returned by the solver. However, if the solver fails to solve the formula, it indicates that the path computed by FINDCFPATH is infeasible. We need to backtrack (BACKTRACK) and explore an alternative path.

On meeting infeasibility this version of SymTest backtracks by potentially multiple decision edges in $stack$. The decision edge b to which this backtracking takes place is computed by BTP. BTP function takes as a parameter $W : E(\mathcal{G}) \times E(\mathcal{G}) \rightarrow \mathbb{R}$. W is a map which takes takes two edges e_1 and e_2 in $E(\mathcal{G})$ as inputs and returns a weight that is related with the probability of computing a *short* test path that reaches e_2 through e_1. The BTP function itself is presented in Algorithm 2, after introducing the prerequisite matter about the preprocessing steps.

3.2 Weighted Backtracking Strategy

An overview of WBS architecture is given in Fig 2. The preprocessing and computing backtrack point are the major steps of WBS. In preprocessing, the experience about the program under test is collected through previous runs. Note that once this preprocessing step is done, its output is usable by any number of runs of SymTest. After preprocessing, backtracking point is computed using the pending targets to be covered and edge ordering.

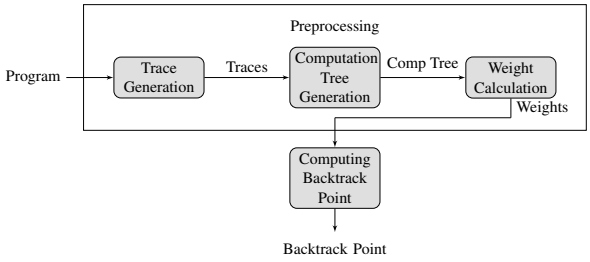


Figure 2: WBS Architecture

3.2.1 Trace Generation

We define a trace π as a list of decision edges $\{e_1, e_2, ..., e_n\}$ which are traversed during a particular run of the program. To generate traces, we instrument the program under test such that every time a edge is traversed during its execution, its corresponding edge id is printed into a file. Let D be the pre-assigned maximum depth of execution. This instrumented program is executed N number of times to generate the set of traces $\Pi = \{\pi_1, \pi_2, ..., \pi_N\}$. These are used to generate the computation tree.

3.2.2 Computation Tree Generation

A computation tree is the compressed representation of set of all the traces (Π). The computation tree is a suffix tree defined over the collected traces Π. Let ($treepath(\pi_i)$) be the path in computation tree ($\mathcal{T}$) which corresponds to the trace π_i in Π. If two traces π_1 and π_2 have a common prefix, this will result in their corresponding paths ($treepath(\pi_1)$ and $treepath(\pi_2)$ respectively) in the computation tree to be merged from the root till the point where π_1 and π_2 diverge. The computation tree CT has a unique root node denoted by $CT.root$. Each node $\mathbf{n}$ in a tree is a triple (e, C, np) where $\mathbf{n}.e$ corresponds to the decision edge in the control flow graph ($\mathcal{G}$) of the program ($\mathcal{P}$), $\mathbf{n}.C$ is the set of child nodes of $\mathbf{n}$ and $\mathbf{n}.np$ is the number of individual tree paths in which $\mathbf{n}$ occurs.

3.2.3 Weight Calculation

The selection of backtracking point is based on the preferability of a decision edge as a backtracking destination. If the current path traversed is defined by the sequence $\{e_1, e_2, ..., e_n\}$ where we meet unsatisfiability at e_n then the set of *candidate backtracking points* (CBP) are $\{e'_1, e'_2, ..., e'_n\}$ where for a decision edge e_i, $e'_i = flip(e_i)$ is the complimentary decision edge of e_i. Selection of the candidate backtracking points yielding the best result (i.e. the shortest test sequence covering all target edges) is undecidable. Hence, we have devised a heuristic to determine the probably best backtracking point and is computed by calculating a preferability index called *weight* (W) of each CBP. Weight for a CBP is calculated taking into account two metrics: probability of covering all the pending target

edges and length of the resulting test sequence.

Probability Weight (W_P): The probability of reaching a target edge represents how often the target edge is covered by the paths obtained. Let e be the current decision edge, P be the total number of all explored paths passing through e, $T = \{e_{t_1}, e_{t_2}, ...e_{t_n}\}$ be the set of all target edges, $P_{(e,e_{t_i})}$ be the total number of paths that cover the edge e and the target edge e_{t_i} in that order. Then, we estimate the probability to reach target edge e_{t_i} from current decision edge e, which we call the probability weight ($W_P(e, e_{t_i})$), is given by:

$$W_P(e, e_{t_i}) = \frac{P_{(e,e_{t_i})}}{P} \tag{1}$$

Length Weight (W_L) : Shortness of the test sequence is indicated by the length weight (W_L). Shorter the test sequence larger will be the value of W_L. W_L is calculated using individual length weight w_L, ($w_L : V(\mathcal{T}) \times E(\mathcal{G}) \rightarrow \mathbb{R}$). For any treenode $\mathbf{n} \in V(\mathcal{T})$, and target edge $e_{t_i} \in T$, the individual length weight is given as:

$$w_L(\mathbf{n}, e_{t_2}) = \sum_{\mathbf{n}' \in N_2} \frac{\mathbf{n}'.np}{L(\mathbf{n}, \mathbf{n}') + 1} \tag{2}$$

where, $N_2 = \{\mathbf{n}' \in V(\mathcal{T}) | \mathbf{n}'.e = e_{t_2}\}$, $\mathbf{n}'$ is reachable from $\mathbf{n}$, np is defined in Section 3.2.2, $L(\mathbf{n}, \mathbf{n}')$ is the length of the path from $\mathbf{n}$ and $\mathbf{n}'$ measured as the number of occurrences of the true branch of the branching node representing the outer loop.

Length weight, W_L of a decision edge $e \in E(\mathcal{G})$ is the sum of the individual length weights of each treenode $\mathbf{n} \in V(\mathcal{T})$.

$$W_L(e_1, e_{t_2}) = \sum_{\mathbf{n} \in N_1} w_L(\mathbf{n}, e_{t_2}) \tag{3}$$

where, $N_1 = \{\mathbf{n} \in V(\mathcal{T}) | \mathbf{n}.e = e_1\}$

Individual Weight (W_I): Individual weight $W_I(e', e_{t_i})$ is defined as the weighted sum of the probability weight and length weight between edges e' in CBP and e_{t_i} in target edge set. Thus:

$$W_I(e', e_{t_i}) = W_P(e', e_{t_i}) + W_L(e', e_{t_i}) \tag{4}$$

Composite Weight (W): Composite weight $W(e')$ of a decision edge e' is the cumulative weight of e' for all target edges given by:

$$W(e') = \sum_{e_{t_i} \in T'} W_I(e', e_{t_i}) \tag{5}$$

where T' is the pending target edge set(defined in Section 3.2.4) for e' with program control state defined by $stack$.

3.2.4 Computing Backtracking Point

At any point during the symbolic execution, the control state of the execution is captured by the state of $stack$. Some of the edges in the target edge set T may be included in this $stack$. These target edges do not need to be covered in the further part of the test sequence. The remainder of the target edges are the *pending targets* (T'), which need to be covered in the further part of the generated test sequence. $T'(e_i{}')$ denotes the pending target edges to be covered from $e_i{}'$.

Consider CBPs e'_1 and e'_2 such that $W(e'_1) > W(e'_2)$. We may want to select e'_1 as the final backtracking point. However, this may turn out to be a mistake. Let $T'(e'_1) = \{e^1_{t_1}, e^1_{t_2}\}$ and $T'(e'_2) = \{e^2_{t_1}, e^2_{t_2}\}$ denote the set of pending target edges for e'_1 and e'_2 respectively. The test sequence to be computed for e'_1 would have a suffix $ts^1_1 = \langle e'_1, e^1_{t_1}, e^1_{t_2}\rangle$ or $ts^1_2 = \langle e'_1, e^1_{t_2}, e^1_{t_1}\rangle$; likewise, for e'_2, they will be $ts^2_1 = \langle e'_2, e^2_{t_1}, e^2_{t_2}\rangle$ or $ts^2_2 = \langle e'_2, e^2_{t_2}, e^2_{t_1}\rangle$. Whether e'_1 should be finally chosen as the backtracking point, or e'_2, depends on which of ts^1_1, ts^1_2, ts^2_1 and ts^2_2 turn out to be the best path. For example, if the weights of the path suffixes ts^1_1, ts^1_2,ts^2_1, ts^2_2 (path weight will be defined below) turn out to be related in the following manner: $W(ts^2_1) > W(ts^2_2) > W(ts^1_1) > W(ts^1_2)$, it is then better to select e'_2 as the backtracking point inspite of the fact that $W(e'_1) > W(e'_2)$. In summary, to make a decision on the backtracking point, we must not just consider the weight of a CBP, but also the weight of the path that will be followed thereafter. In order to achieve this, *edge ordering* of targets edges is performed.

Unfortunately, for a given CBP e'_i, problem of computing the best test sequence suffix $\langle e'_i, ts^i_1, ts^i_2, ..., ts^i_n\rangle$ is akin to *vehicle routing problem* (VRP) A brute force approach would compute the path weights corresponding to all permutations of the pending target edges T' which is exponential in $|T'|$. For an arbitrarily large T', this would be too expensive.

Algorithm 2 Computing Backtracking Point

```
1: function BTP(G, stack, W, T)
2:     s ←< flip(e)∀(e, tf) ∈ stack >
3:     D' ← the subset of s such that it consists of upto
       N elements of s whose composed weights are the max-
       imum.
4:     for all d ∈ D' do
5:         T' ← PENDINGTARGETS(d, T, stack)
6:         EO[d] ← EDGEORDERING(d, T')
7:     return argmin_{d∈D'} EO[d]
```

In order to perform target edge ordering, initially a *weighted graph*($\mathcal{G}_W$) is created using a given set of pending target edges T' in $E(\mathcal{G})$. $\mathcal{G}_W$ is a complete weighted digraph such that:

- The node set $V(\mathcal{G}_W)$ corresponds to the edge of pending target edge set T'. $\forall n \in V(\mathcal{G}_W)$, $n.e$ denotes the target edge corresponding to it.
- $\forall n_i, n_j \in V(\mathcal{G}_W)$,

$$n_i \rightarrow_{G_W} n_j \text{ if } W(n_i.e, n_j.e) \geq W(n_j.e, n_i.e)$$
$$n_i \rightarrow_{G_W} n_j \text{ otherwise}$$

  where $n_i \rightarrow_{G_W} n_j$ means that there exists an edge between n_i and n_j.
- An edge $e(n_i, n_j)$ from a node n_i to n_j is annotated by a number $e(n_i, n_j).w$ given by:

$$e(n_i, n_j).w = \frac{1}{W(n_i.e, n_j.e)}$$

Augmented Weighted Graph ($\mathcal{G}'_W$) The weighted graph is augmented with the candidate backtracking point to get $\mathcal{G}'_W$. The edge ordering is performed on $\mathcal{G}'_W$. For a given CBP e and weighted graph $\mathcal{G}_W$, We define an augmented weighted graph $\mathcal{G}'_W$ as follows:

- $\mathcal{G}'_W$ has all the nodes and edges of $\mathcal{G}_W$.
- $V(\mathcal{G}'_W) = V(\mathcal{G}_W) \cup \{n\}$ such that $n.e = e$.
- For all nodes $n' \in V(\mathcal{G}'_W) \setminus \{n\}$, there exists an edge $e(n, n')$ such that $e(n, n').w = \frac{1}{W(n.e, n'.e)}$.

Algorithm 2 presents BTP function that selects a decision edge from the candidate backtrack points (CBP) after performing edge ordering on the pending targets edges.

4 Experimental Evaluation

In this section we first describe the experimental set up. We then explain the benchmarks used to evaluate the implementation followed by a discussion of the results.

4.1 Set up

We have implemented WBS and integrated it with SymTest. The weight is calculated using 500 previous runs of the program under test. To evaluate the effectiveness of SymTest-WBS, we considered the following factors:

- time taken for test sequence generation
- test sequence length i.e., the number of iterations taken to cover the target edges
- number of execution paths processed to generate the test sequence covering the target edges

A large number of execution paths processed for test sequence generation will result in an increase in the time for test sequence generation. More number of iterations to cover the target edge increases interaction with the external environment. Both these facts contribute to the cost of testing of an embedded software.

We compared SymTest-WBS with following tools on the basis of above parameters:

- SymTest [1]: SymTest framework where backtracking proceeds by a single node.
- KLEE [9]: Symbolic execution tool build over LLVM, that generates test cases with high code coverage.

Each of the above mentioned tools is evaluated against different test programs. We ran our tool on the benchmark programs on an Intel Quad Core i7-3770 3.40GHz machine running Ubuntu 18.04.4.

The number of iterations of the main loop is considered as the performance metric for measuring the effectiveness of symbolic execution search strategies[10]. For all our experiments we have set the maximum depth of execution (i.e. number of iterations of the main loop) as 5.

4.2 Benchmarks

TABLE 1 shows the list of benchmark programs that we used in our experiments and the results obtained. We used the benchmark programs which are used by other symbolic execution tools [11] [12] [1]. As we are concentrating on embedded software, apart from available benchmarks, we have converted a set of single task programmable logic control programs [13] into its equivalent C program. The other test programs are from SVCOMP test suite [14], CREST test programs [12], RERS challenge [15].

4.3 Results

TABLE 1 shows the results of experiments. The "#iter" column gives the number of iterations taken by the generated test case to cover the target edges,"#path" column gives the number of explored paths, "Target Cov" gives the percentage of the target edges covered. The "Time (ms)" column gives the time taken by the tool in milliseconds. The total number of iterations is calculated by taking the sum of iterations taken by each test data. A time out (TO) happens at a time limit of 600000ms. A time out means that the tool could not generate a test sequence that covers all the target edges within the stipulated time-out period.

On comparing the proposed approach with SymTest, It is observed that in test_program, cfg1 SymTest-WBS produces shorter test sequence length in less time compared to SymTest. In problem1M, cfg2 SymTest timed out, but SymTest-WBS succeeded to cover target edges. On com-

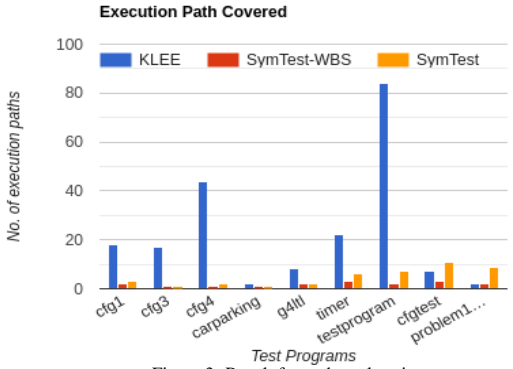


Figure 3: Result for path exploration

Sl No.	Program	SymTest-WBS				SymTest				KLEE			
		#iter	#path	Target Cov	Time(ms)	#iter	#path	Target Cov	Time(ms)	#iter	#path	Target Cov	Time(ms)
1	cfg1	1	2	100	2081.27	2	3	100	2357.11	7	18	100	9126.10
2	cfg2	1	2	100	1040.38	-	-	-	TO	4	14	100	9018.61
3	cfg3	1	1	100	124.10	1	1	100	245.50	3	17	100	8094.71
4	cfg4	1	1	100	337.14	1	2	100	175.77	4	44	100	5804.59
5	test_program	1	2	100	513.50	4	7	100	1001.21	4	84	100	12095.66
6	timer	1	3	100	126.96	1	6	100	234.60	5	22	100	72.26
7	car_parking	1	1	100	117.89	1	1	100	192.36	4	2	100	78.81
8	burglar_alarm	3	2	100	1173.94	3	9	100	5542.12	-	-	-	T0
9	g4ltl1	1	2	100	923.90	1	2	100	1594.23	5	8	100	159.56
10	g4ltl2	1	2	100	923.90	1	2	100	1594.23	5	7	100	163.30
11	trex03	1	1	100	231.54	1	1	100	240.01	1	102	100	1346.62
12	cfg_test	2	3	100	292.42	2	11	100	271.34	2	7	100	380.27
13	problem1M	2	4	100	3819.30	-	-	-	TO	-	-	-	TO
14	problem2M	2	3	100	399.79	2	10	100	8586.02	-	-	-	TO
15	problem11M	2	2	100	510.04	2	9	100	1795.25	2	2	100	73.15
16	problem1	-	-	-	TO	-	-	-	TO	-	-	-	TO
17	problem11	-	-	-	TO	-	-	-	TO	-	-	-	TO

Table 1: Comparison Results

paring the proposed approach with KLEE, it is observed that for programs having majority of statements which are reachable, KLEE takes more time in test sequence generation compared to SymTest and SymTest-WBS for cfg_test, test_program. KLEE focuses on code coverage. If we are interested in complete code coverage (as in first-time testing), KLEE is one of the best choices for test data generation. But if we want to compute test data to reach specific program points which are deep in the code (as often seen in regression testing), then SymTest-WBS performs better. The number of iterations to cover the same target edges is more in KLEE. This can be observed in g4ltl1, timer, car_parking. In problem1M, problem2M KLEE timed out, but SymTest-WBS was successful in generating test sequence.

In problem1, problem11 it is observed that all the three tools timed out. Fig. 3 shows the results of path exploration for those programs which was successful in generating test sequence by all the three tools. SymTest-WBS requires less number of paths to attain target edge coverage compared to SymTest and KLEE. It is worthwhile observing that the number of TO by SymTest-WBS is less compared to SymTest and KLEE. Both SymTest and KLEE explore more number of execution paths to cover target edges. This explains the timeout observed in SymTest and KLEE.

5 Conclusions and Future Work

In this paper we proposed a backtracking heuristic named Weighted Backtracking Strategy. The knowledge about system behaviour through previous runs is exploited to create a computation tree. Using the computation tree the optimal backtracking point is selected. WBS is implemented in SymTest. Experiments shows that the WBS is effective in increasing the set of cases where SymTest achieves termination and has further shortened the test sequence length with respect to those achieved by SymTest with simple backtracking. Our future work involves using machine learning techniques for selecting backtracking point.

References

[1] S. Chakrabarti and R. S, "Symtest a framework for symbolic testing of embedded software," in *SymTest*. ISEC, 2016.

[2] P. Godefroid, N. Klarlund, and K. Sen, "DART: directed automated random testing," in *PLDI*, 2005. [Online]. Available: http://doi.acm.org/10.1145/1065010.1065036

[3] *CUTE: a concolic unit testing engine for C*. 13th ACM SIGSOFT international symposium on Foundations of software engineering, ESEC/FSE-13, 2005.

[4] S. Anand, C. S. Păsăreanu, and W. Visser, "Jpf-se: A symbolic execution extension to java pathfinder," in *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 134–138. [Online]. Available: http://dl.acm.org/citation.cfm?id=1763507.1763523

[5] *Symbolic PathFinder: integrating symbolic execution with model checking for Java bytecode analysis*. Automated Software Engineering, 2013.

[6] M. Kim, Y. Kim, and Y. Jang, "Industrial application of concolic testing on embedded software: Case studies," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, April 2012, pp. 390–399.

[7] Y. Kim, M. Kim, Y. J. Kim, and Y. Jang, "Industrial application of concolic testing approach: A case study on libexif by using crest-bv and klee," *2012 34th International Conference on Software Engineering (ICSE)*, pp. 1143–1152, 2012.

[8] J. Burnim and K. Sen, "Heuristics for scalable dynamic test generation," in *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 443–446. [Online]. Available: http://dx.doi.org/10.1109/ASE.2008.69

[9] C. Cadar, D. Dunbar, and D. Engler, "Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 209–224. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855741.1855756

[10] T. Xie, N. Tillmann, J. de Halleux, and W. Schulte, "Fitness-guided path exploration in dynamic symbolic execution," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, June 2009, pp. 359–368.

[11] J. Jaffar, R. Maghareh, S. Godboley, and X.-L. Ha, "Tracerx: Dynamic symbolic execution with interpolation (competition contribution)," in *Fundamental Approaches to Software Engineering*, H. Wehrheim and J. Cabot, Eds. Cham: Springer International Publishing, 2020, pp. 530–534.

[12] J. Burnim and K. Sen, "Heuristics for scalable dynamic test generation," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, 2008, pp. 443–446.

[13] https://www.sanfoundry.com, [Online; accessed 19-September-2020].

[14] https://sv-comp.sosy-lab.org/2021/benchmarks.php, [Online; accessed 19-September-2021].

[15] http://rers-challenge.org/2017/index.php, [Online; accessed 10-September-2021].

An Exploratory Study of Bug Prioritization and Severity Prediction based on Source Code Features

Chun Ying Zhou
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
zcy9838@stu.hubu.edu.cn

Cheng Zeng
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
zc@hubu.edu.cn

Peng He*
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
penghe@hubu.edu.cn

***Abstract*—Software systems generate a large number of bugs during their lifecycles. Managing and assigning these bug reports is a challenging task. Building prediction models for the priority or severity levels of bugs through bug reports can help developers prioritize highly urgent bugs. Traditional prediction models are based on the textual description information in bug reports. However, most of the description is little or no. According to the bug report, developers need to fix the corresponding source code files. If the corresponding source code file is a core module in a software system, the report is likely to have high-level assignment rights. Therefore, in this paper, we investigate the effect of using the source code file feature sets on classification performance. In addition, we evaluate the effect of different sampling methods on the data, namely SMOTE, RUS, SMOTEEN, Adaboost, and GAN. Extensive experiments were conducted on five open-source projects. The experimental results show that the source code file feature sets do not perform as well as the textual description features in bug reports. Besides, over-sampling methods do not alleviate the data imbalance problem in the case of insufficient data, while GAN performs best in the case of sufficient data.**

***Keywords*—bug priority; bug severity; bug report; source code; data imbalance**

I. INTRODUCTION

Throughout the software lifecycle, developers will receive a large number of bug reports. Once a bug report is committed and confirmed, the bug needs to be fixed in a timely manner. Different analyses can be performed based on the bug reports, including Bug Triage, Bug Localization, Bug-fix Time Estimation, predict defect attributes etc [1]. A bug report contains several attributes, such as *bug id*, *summary*, *description*, *reporter*, *created date*, *fix version*, *status*, *priority*, *severity*, in which the priority levels range from P1 (most important) to P5 (least important), and the severity is categorized into *blocker*, *critical*, *major*, *normal*, *minor*, *enhancement*, and *trivial*. Priority is assigned from the developer's point of view, which indicates the urgency of fixing bugs. Severity is assigned from the user's point of view, which indicates the degree of impact on the use of software function [2].

Existing studies analyzed the text of bug reports by using machine learning methods to automatically predict priority or severity [3]. Given the excellence of deep learning in the field of natural language processing, researchers also explored various neural networks to further extract semantic information from bug reports [1, 6-9]. Unfortunately, if the bug reports provide insufficient or misleading information, the performance of the predictor will be greatly affected. Therefore, in addition to the textual features of bug reports, whether there are other appropriate features for bug prioritization and severity prediction becomes an open challenge. To this end, we will further consider the information of source code files. We assume that if a source code file is a core file in the project, then it has a higher level of importance. Once the file is defective, it is more likely to have a higher priority or severity. Leveraging neural networks to capture semantic and syntactic features from source files is widely used for bug localization [4] and defect prediction [5].

In addition, during bug repair, the co-change relationships between source files have been proved to be potentially valuable [10]. We assume that if two files are associated with a bug report at the same time, there will be some correlation between these two files. More occurrences together indicate a stronger relationship. Hence, we construct a co-occurrence network (COON) between source code files, and extract relational features of the source files by network embedding learning.

The study aims at investigating the feasibility of source code feature sets mentioned above on bug prioritization and severity prediction, and analyzing the impact of different features on this task. To simplify the subsequent presentation, we use *FSet-1* to represent the feature set learned from the textual information of the bug reports, *FSet-2* to represent the semantic feature set of the source code files associated with the bug reports, and *FSet-3* to represent the relational feature set learned from the co-occurrence network of the source code files. Extensive experiments were conducted on five open-source projects to answer the following research questions:

RQ1: Which type of feature set performs better?

RQ2: Is it helpful to consider the importance of source code files?

RQ3: Do the combined feature sets achieve better results?

RQ4: Is the impact of sampling methods obvious?

The remainder of this paper is organized as follows. The related work is presented in Section II. The method is detailed in Section III. The evaluation and analysis are presented in Section IV. Finally, the conclusion is drawn in Section V.

II. RELATED WORK

Most of the existing models predict various attributes based on textual analysis of bug reports (e.g., summary and

Corresponding author: Peng He (penghe@hubu.edu.cn)

DOI reference number: 10.18293/SEKE2022-102

description). Tian et al. [2] treated priority prediction as a linear regression problem rather than a classification problem. The priority level is an ordinal value rather than a categorical value, while classification will make a large difference between levels. Sharma et al. [3] evaluated the performance of different machine learning techniques such as SVM, Naive Bayes, and K-Nearest Neighbors in the priority prediction. Kumari et al. [6] took care of uncertainty by using entropy-based measures. Umer et al. [7] and Ramay et al. [8] proposed an automatic emotion-based prediction method for sentiment analysis of bug reports. Bani-Salameh et al. [9] used a five-layer RNN-LSTM neural network model for bug priority prediction.

If the reporter provided an insufficient description of the bug, it is difficult to learn valuable features from it. As far as we know, few studies have explored bug prediction and severity in terms of the feature sets of source code. In fact, a bug report is often associated with at least one source code file that needs to be fixed. Source code files can be used to indirectly learn about potential bug features. The source code files are converted into Abstract Syntax Trees (ASTs). Then a tree-based neural network is used to extract semantic information. Compared to the source code, the ASTs ignore unnecessary details but still retain lexical and syntactic structure information [11].

With the wide use of complex networks in software engineering, a co-occurrence network is proposed in this paper inspired by this research trend. Considering that source code files are not independent, when two files are associated with a bug report at the same time, there is an association relationship between them, and a complex network is constructed based on this correlation. In fact, real-world networks are often more complex in that not only topological information is available, but also each node and edge has attributes. However, traditional network embedding methods are unsupervised and cannot utilize node attribute information. With the development of Graph Neural Networks (GNNs), a group of models has emerged specifically for learning graph structure data [12], which can smoothly incorporate node and edge attributes while learning the network structure to generate better robust representation.

Unlike existing studies, we not only perform textual analysis of bug reports, but also consider semantic information of source code files and relational information based on co-occurrence network. Therefore, the feature sets in this paper are divided into three categories: textual features of bug reports; semantic features of source code files; and relational features of co-occurrence networks. These three sets of features are then combined and applied to bug prioritization and severity prediction.

III. APPROACH

The workflow (cf. Figure. 1) comprises three parts: (1) *FSet-1* generation: extract the summary and description of bug reports, then use CNN to learn the textual features; (2) *FSet-2* generation: convert source code files to ASTs, then use CNN to extract the code semantic features; (3) *FSet-3* generation: construct a co-occurrence network between source code files, then use GNN to learn the structural features. Finally, the three feature sets are combined in different ways and fed into the classifier for prediction.

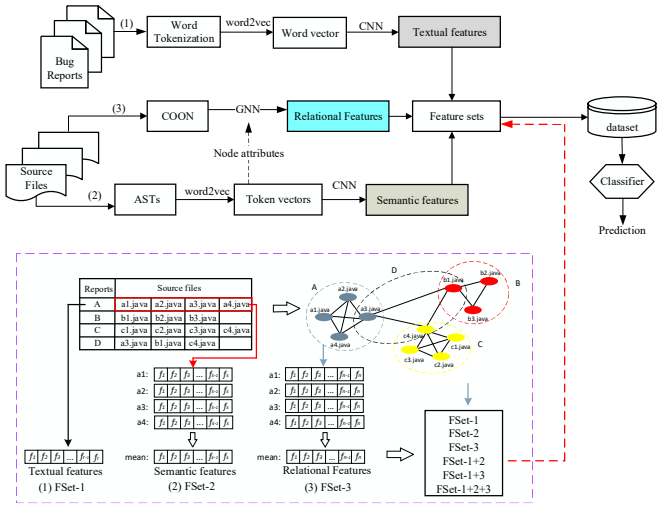


Figure 1. Experimental workflow.

A. FSet-1 generation

The summary and description of each bug report are combined into a document, then all textual information is preprocessed. First, tokenize each word in the text sequences. Second, remove words without real meaning, such as "the", "and", "this", "that", etc. Finally, stem each word into basic words. For example, "working" and "worked" will be converted into "work". In addition, we further processed the tokens as treated in [4]. Since developers usually use compound words to name classes and methods. Therefore, according to CamelCase naming rules, compound words are split into separate real words. For example, "WorkbenchActionBuilder" is split into "Workbench", "Action" and "Builder". After preprocessing, each token is converted into a word vector using *word2vec* and then fed into CNN to extract textual features for bug reports.

B. FSet-2 generation

In this part, the open-source python package javalang[1] is adopted to parse the source code files into ASTs. Three types of nodes were selected as in [13]: (1) nodes of method invocations and class instance creations, (2) declaration nodes, and (3) control-flow nodes. After parsing, each Java file is converted into a token sequence. Since CNN requires input integer vectors, each token is mapped to a unique integer. That is, the token sequence is converted into an integer vector. Since the length of the token sequence of each file is unequal, the dimensions of the converted integer vectors will be different. To keep the same dimension of each file vector, 0 is added at the end of the integer vector, which is equal to the length of the longest vector. Note that adding 0 will not affect the result. Moreover, some uncommon tokens are filtered out and only the tokens that appear more than three times are encoded.

Suppose that there are n Java source code files associated with all bug reports for project P, $P = \{f_1, f_2, \dots, f_n\}$. After the above processing, the token sequence $f_i = \{t_{i1}, t_{i2}, \dots, t_{im}\}$ is extracted, then each token is mapped to an integer, i.e., f_i is converted to a fixed-length integer vector $f_i \in \mathbb{R}^l, i \in [1, n]$. In the embedding layer, f_i is converted to a real-valued vector matrix $X_i = \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \dots, \boldsymbol{x}_{il}\}, X_i \in \mathbb{R}^{l \times d}$, where $\boldsymbol{x}_{ij} \in \mathbb{R}^d$ is the embedding vector corresponding to the j-th token t_{ij} of f_i.

[1] https://github.com/c2nes/javalang

Since the source code files associated with the bug reports are defective files, clean source code files are also needed to be fed into the CNN for training together as positive instances. Therefore, positive instances are randomly selected from the remaining files in the project, keeping the same number of negative instances. After convolution and pooling layers, the *FSet-2* is generated.

C. FSet-3 generation

Figure 1 also shows the construction of a co-occurrence network through a simple example with four bug reports A, B, C, and D. The source code files associated with these reports are listed on the right. Clearly, for A, B, and C, their associated source code files form three fully connected communities respectively, while the associated files of D connect them, thus forming a large co-occurrence network (COON). In COON, two files may co-occur frequently, so in order to distinguish the strength of the relationship between files, we use the co-occurrence times as a weight. Note that the co-occurrence relationship is undirected in our context.

After constructing COON, the relational features are learned by using a GNN model, which iteratively updates the representation of each node by aggregating the features of its neighboring nodes. This process is mainly divided into two steps. First, aggregate the features of neighboring information to obtain $a_v^{(k)}$, and then combine the neighboring features $a_v^{(k)}$ with the node features of the previous layer $h_v^{(k-1)}$, in order to obtain the updated features.

$$a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)}: u \in N(v)\}) \quad (1)$$

$$h_v^{(k)} = COMBINE^{(k)}(h_v^{(k-1)}, a_v^{(k)}) \quad (2)$$

where $h_v^{(k)}$ is the vector of node v at the k-th layer. $h_v^{(0)} = X_v$. $N(v)$ is the set of neighbors of node v. $AGGREGATE(\cdot)$ and $COMBINE(\cdot)$ are the aggregation function and combination function respectively.

Given that the source code files are defective, and when there is only one class of instance labels, it is not possible to use a supervised model. However, adding additional instances would change the structure of the co-occurrence network. Therefore, we choose an unsupervised model DGI [14] to learn COON. Generating this type of feature set mainly consists of three steps: (1) constructing the network; (2) initializing the node attributes; and (3) using DGI to capture the topological structure information and generate relational features. DGI can learn both network relational and node attribute features. Since the original COON has no node attributes, the node attributes in the COON should be provided before training the DGI. In addition, rich node attributes allow DGI to be better trained. Consequently, the token vectors extracted from the source code are used as the initial node attributes. Finally, the network topology and node attributes are fed into DGI, and *FSet-3* is output.

IV. Experiments and Analysis

A. Datasets

In this paper, we use five public datasets commonly used in Bug Localization[2]. We queried the bug priority and severity attributes from https://bugs.eclipse.org/bugs/ and https://bz.apache.org/bugzilla/ to label these datasets. It is well known that data imbalance is one of the problems in multi-class prediction. To accurately predict the labels for each class, a large amount of data is needed for training. The majority labels of bug reports are P3 or *normal*, which cannot be learned adequately for minor classes. Therefore, priority and severity levels are coarse-grained into three categories. For priority, P1 and P2 are classified as *high*, P3 as *medium*, P4 and P5 as *low*. For severity, *blocker*, *critical* and *major* are classified as *severe*, *normal* as *normal*, *minor*, *enhancement* and *trivial* as *non-severe*.

Table I gives the statistics of the datasets. It can be seen that *medium* and *normal* are the majority categories in most projects. The data imbalance problem of priority prediction is particularly serious. For better empirical validation, we dropped the *medium* category from the priority and the *normal* category from the severity. That is, we perform binary classification aimed to help developers prioritize fixing high-level bugs.

TABLE I. Summary of the Datasets

Products	Bug priority prediction			Bug severity prediction		
	high	***medium***	***low***	***severe***	***normal***	***non-severe***
AspectJ	107	477	9	114	378	101
Eclipse	1072	5301	122	1235	4473	787
JDT	1020	5163	91	690	4410	1174
SWT	230	3901	20	787	3090	274
Tomcat	981	53	22	114	636	306

B. Settings

A five-fold cross-validation is used. For each run, the dataset is divided into five copies, of which 80% is used for training and 20% for testing, with each division ensuring that the ratio of positive to negative instances is approximated. For each project, the experiment was repeated 25 times, and the final results were averaged to reduce the bias introduced by dividing the data randomly.

The output dimension of three feature sets is set to 16. For the CNN used in *FSet-1* and *FSet-2*, a four-layer architecture is employed, including an embedding layer, a convolutional layer, a max-pooling layer and a fully connected layer. The batch size is set to 32 and the epoch is 100, using Adam optimizer with a learning rate of 0.001. For the DGI used in *FSet-3*, a two-layer convolution is employed. The dimension of the hidden layer is set to 64. The DGI is a full batch training with epochs of 200, also using the Adam optimizer with a learning rate of 0.001. The classifier is MLP. The F-measure and Accuracy are used as evaluation metrics in this paper.

C. Analysis

1) RQ1: Which type of feature set performs better?

FSet-1 is the textual features for bug reports. For better illustration, the preprocessed text is treated as the original feature without CNN learning, namely *origin*. Tables II and III show the prediction results. It can be seen that *FSet-2* and *FSet-3* perform poorly, even worse than the *origin*. For priority prediction, due to the extreme imbalance of the datasets, both *FSet-2* and *FSet-3* predict into the majority class labels, which could not construct a reasonable prediction model. For severity prediction, the results are similar to the priority prediction.

[2] https://github.com/yanxiao6/BugLocalization-dataset

Moreover, the performance of *FSet-2* is much worse than that of *FSet-3*.

2) **RQ2: Is it helpful to consider the importance of source code files?**

Based on our assumption, if the source code file is a core file with a high importance level in the project, then the bug reports associated with it are likely to be of high priority or severity level. Since the projects in the dataset used in this paper are not from the same version, and many of them are test files, which cannot be found in the official released version. Therefore, we can only calculate the importance of the source code files from the available data. Specifically, if the corresponding bug report has a *high* or *severe* label, the importance value of all source code files associated with it will be added by 10, while the *medium* and *normal* will be added by 3, and the *low* and *non-severe* will be added by 1. As shown in Figure 2, the importance value of the source code file a.java associated with bug reports A, C and D is 3+1+3=7. Similarly, after calculating the importance values of all source code files, the importance of the bug reports can be obtained indirectly. For example, the importance of bug report A is the sum of the importance values of a.java, b.java, c.java, and d.java. Tables IV and V show the results of priority and severity prediction. △ represents the improvement considering the importance of source code files.

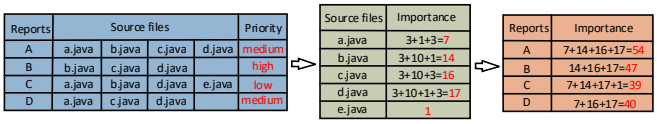


Figure 2. An example of importance calculation.

TABLE II. THE PERFORMANCE OF FEATURE SETS ON PRIORITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
origin	F1	0.518	0.506	0.518	**0.515**	0.496	0.511
	ACC	0.874	0.826	0.853	0.870	0.965	0.878
FSet-1	F1	**0.804**	**0.812**	**0.802**	0.479	**0.770**	**0.734**
	ACC	**0.966**	**0.940**	**0.949**	**0.920**	**0.987**	**0.952**
FSet-2	F1	0.480	0.473	0.479	0.479	0.494	0.481
	ACC	0.922	0.898	0.918	**0.920**	0.978	0.927
FSet-3	F1	0.480	0.473	0.479	0.479	0.494	0.481
	ACC	0.922	0.898	0.918	**0.920**	0.978	0.927

TABLE III. THE PERFORMANCE OF FEATURE SETS ON SEVERITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
origin	F1	0.606	0.560	0.592	0.537	0.555	0.570
	ACC	0.612	0.583	0.617	0.638	0.658	0.622
FSet-1	F1	**0.893**	**0.848**	**0.859**	**0.883**	**0.840**	**0.865**
	ACC	**0.894**	**0.856**	**0.872**	**0.913**	**0.877**	**0.882**
FSet-2	F1	0.398	0.390	0.428	0.427	0.421	0.413
	ACC	0.514	0.610	0.629	0.742	0.729	0.645
FSet-3	F1	0.630	0.532	0.521	0.485	0.481	0.530
	ACC	0.644	0.630	0.648	0.751	0.722	0.679

In priority prediction, using the importance feature for *FSet-1* reduced the overall prediction performance. In particular, the negative impact was significant for *AspectJ* and *Tomcat*, and *SWT* predicted all the labels into the majority class. Due to the relatively sufficient data, the impact on *Eclipse* and *JDT* was slight. For *FSet-2, AspectJ*, *SWT* and *Tomcat,* with insufficient data, still failed to build a reasonable model. For *Eclipse* and *JDT*, on the other hand, achieved some improvements. The impact of importance feature for *FSet-3* was not positive. In severity prediction, using the importance feature for *FSet-1* had a significant negative impact on *AspectJ*, indicated by -32.4% F1 value, but the impact on other projects was not significant. The importance feature showed a great improvement for *FSet-2*, while it had little impact for *FSet-3*.

In short, the results show that the introduction of the importance feature of the source code files does not benefit *FSet-1* and *FSet-3*, and even reduces the prediction performance, while it is a great improvement for *FSet-2*. In addition, based on the use of the source file importance feature, the performance of *FSet-2* is better than that of *FSet-3*. Moreover, the problems of data imbalance and data insufficiency have a great impact on prediction. Data imbalance leads to difficulties in constructing reasonable model, while data insufficiency leads to large fluctuations in prediction results.

TABLE IV. THE PERFORMANCE OF USING IMPORTANCE ON PRIORITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
FSet-1	F1	0.480	**0.829**	0.780	0.479	0.572	0.628
	△	-32.4%	**1.8%**	-2.2%	0.0%	-19.8%	-10.6%
	ACC	0.922	**0.947**	0.947	0.920	0.980	0.943
	△	-4.3%	**0.7%**	-0.3%	0.0%	-0.7%	-0.9%
FSet-2	F1	0.480	**0.579**	**0.514**	0.479	0.494	**0.509**
	△	0.0%	**10.6%**	**3.5%**	0.0%	0.0%	**2.8%**
	ACC	0.922	**0.908**	0.921	0.920	0.978	**0.930**
	△	0.0%	**1.0%**	**0.3%**	0.0%	0.0%	**0.3%**
FSet-3	F1	0.480	0.472	0.476	0.479	0.494	0.480
	△	0.0%	-0.1%	-0.2%	0.0%	0.0%	-0.1%
	ACC	0.922	0.860	0.862	0.920	0.977	0.908
	△	0.0%	-3.8%	-5.6%	0.0%	-0.1%	-1.9%

TABLE V. THE PERFORMANCE OF USING IMPORTANCE ON SEVERITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
FSet-1	F1	0.719	0.845	**0.863**	0.879	**0.860**	0.833
	△	-17.4%	-0.2%	**0.4%**	-0.4%	**2.0%**	-3.2%
	ACC	0.751	0.854	**0.875**	0.910	**0.891**	0.856
	△	-14.3%	-0.1%	**0.3%**	-0.4%	**1.4%**	-2.6%
FSet-2	F1	**0.656**	**0.616**	**0.542**	**0.606**	**0.576**	**0.599**
	△	**25.9%**	**22.6%**	**11.4%**	**17.9%**	**15.5%**	**18.7%**
	ACC	**0.660**	**0.686**	**0.658**	**0.774**	**0.731**	**0.702**
	△	**14.5%**	**7.6%**	**2.9%**	**3.2%**	**0.2%**	**5.7%**
FSet-3	F1	0.526	**0.553**	0.508	**0.525**	**0.588**	**0.540**
	△	-10.3%	**2.1%**	-1.3%	**4.0%**	**10.7%**	**1.0%**
	ACC	0.564	0.625	0.586	0.616	**0.736**	0.625
	△	-8.0%	-0.5%	-6.2%	-13.5%	**1.4%**	-5.4%

3) **RQ3: Do the combined feature sets achieve better results?**

According to the results obtained above, *FSet-2* and *FSet-3* are much less effective than *FSet-1*. Inspired by this, can hybrid feature sets further improve prediction performance? In this RQ, we explore three combinations: *FSet-1+2*, *FSet-1+3* and *FSet-1+2+3*. For example, given a bug report A, the source code files associated with it are $a_1.java$, $a_2.java$, $a_3.java$ and $a_4.java$, where the importance of the source code files are imp_1, imp_2 imp_3, and imp_4. The dimension of *FSet-1* is d_1, while *FSet-2* is d_2 and *FSet-3* is d_3. Then the combined features of A are respectively expressed as follows:

$$\boldsymbol{x_r} = [x_1, x_2, \dots, x_{d_1}] \quad (6)$$

$$\boldsymbol{x_s} = [x_1, x_2, \dots, x_{d_2}] \quad (7)$$

$$\boldsymbol{x_n} = [x_1, x_2, \dots, x_{d_3}] \quad (8)$$

$$\boldsymbol{x}_{\boldsymbol{c_{1+2}}} = \boldsymbol{x_r} \oplus mean(imp_1 \cdot \boldsymbol{x_{s_1}}, imp_2 \cdot \boldsymbol{x_{s_2}}, imp_3 \cdot \boldsymbol{x_{s_3}}, imp_4 \cdot \boldsymbol{x_{s_4}}) \quad (9)$$

$$\boldsymbol{x}_{\boldsymbol{c_{1+3}}} = \boldsymbol{x_r} \oplus mean(imp_1 \cdot \boldsymbol{x_{n_1}}, imp_2 \cdot \boldsymbol{x_{n_2}}, imp_3 \cdot \boldsymbol{x_{n_3}}, imp_4 \cdot \boldsymbol{x_{n_4}}) \quad (10)$$

$$\boldsymbol{x}_{\boldsymbol{c_{1+2+3}}} = \boldsymbol{x_r} \oplus mean\big(imp_1 \cdot \boldsymbol{x_{s_1}}, imp_2 \cdot \boldsymbol{x_{s_2}}, imp_3 \cdot \boldsymbol{x_{s_3}}, imp_4 \cdot \boldsymbol{x_{s_4}}\big) \oplus mean(imp_1 \cdot \boldsymbol{x_{n_1}}, imp_2 \cdot \boldsymbol{x_{n_2}}, imp_3 \cdot \boldsymbol{x_{n_3}}, imp_4 \cdot \boldsymbol{x_{n_4}}) \quad (11)$$

where $\boldsymbol{x_r} \in \mathbb{R}^{d_1}$ is *FSet-1*, $\boldsymbol{x_s} \in \mathbb{R}^{d_2}$ is *FSet-2*, $\boldsymbol{x_n} \in \mathbb{R}^{d_3}$ is *FSet-3*, $\boldsymbol{x}_{\boldsymbol{c_{1+2}}} \in \mathbb{R}^{d_1+d_2}$ is *FSet-1+2*, $\boldsymbol{x}_{\boldsymbol{c_{1+3}}} \in \mathbb{R}^{d_1+d_3}$ is *FSet-1+3*, $\boldsymbol{x}_{\boldsymbol{c_{1+2+3}}} \in \mathbb{R}^{d_1+d_2+d_3}$ is *FSet-1+2+3*. $mean(\cdot)$ is the mean function, $\oplus$ is feature splicing symbol.

Tables VI and VII show the results of priority prediction and severity prediction. △ shows an improvement compared to *FSet-1*. The results show that the combined features are not as effective as expected. *FSet-1+3* and *FSet-1+2+3* were decreased greatly, indicating that *FSet-3* was highly destructive to *FSet-1*. Although *FSet-1+2* decreased slightly and can improve prediction performance in some cases, the enhancement was not significant. Hence, blindly combining feature set for bug prediction can easily lead to negative effects.

TABLE VI. THE PERFORMANCE OF COMBINED FEATURE SETS ON PRIORITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
FSet-1+2	F1	0.679	**0.815**	0.781	0.479	0.581	0.667
	△	-12.5%	**0.4%**	-2.2%	0.0%	-18.9%	-6.7%
	ACC	0.945	0.939	0.945	0.920	0.978	0.945
	△	-2.1%	-0.1%	-0.4%	0.0%	-0.9%	-0.7%
FSet-1+3	F1	0.480	0.558	0.518	**0.485**	0.489	0.506
	△	-32.4%	-25.4%	-28.4%	**0.5%**	-28.1%	-22.8%
	ACC	0.922	0.869	0.904	0.886	0.959	0.908
	△	-4.3%	-7.1%	-4.5%	-3.4%	-2.8%	-4.4%
FSet-1+2+3	F1	0.480	0.595	0.532	**0.496**	0.499	0.520
	△	-32.4%	-21.6%	-27.1%	**1.6%**	-27.1%	-21.3%
	ACC	0.922	0.895	0.888	0.850	0.950	0.901
	△	-4.3%	-4.5%	-6.1%	-7.0%	-3.7%	-5.1%

TABLE VII. THE PERFORMANCE OF COMBINED FEATURE SETS ON SEVERITY

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
FSet-1+2	F1	**0.907**	**0.848**	0.857	0.865	0.831	0.862
	△	**1.4%**	**0.04%**	-0.2%	-1.8%	-0.9%	-0.3%
	ACC	**0.908**	**0.856**	0.869	0.899	0.869	0.880
	△	**1.4%**	**0.1%**	-0.2%	-1.4%	-0.9%	-0.2%
FSet-1+3	F1	0.872	0.706	0.788	0.748	0.806	0.784
	△	-2.2%	-14.2%	-7.1%	-13.6%	-3.4%	-8.1%
	ACC	0.873	0.722	0.810	0.822	0.852	0.816
	△	-2.1%	-13.4%	-6.1%	-9.1%	-2.5%	-6.7%
FSet-1+2+3	F1	0.851	0.738	0.775	0.716	0.819	0.780
	△	-4.3%	-11.0%	-8.4%	-16.8%	-2.1%	-8.5%
	ACC	0.860	0.758	0.794	0.777	0.863	0.810
	△	-3.4%	-9.8%	-7.7%	-13.6%	-1.4%	-7.2%

4) **RQ4: Is the impact of sampling methods obvious?**

From Table I, it is clear that the data used for prediction are obviously unbalanced and insufficient. For example, for *SWT,* the ratio of majority class to minority class was as high as 15:1. Extreme data imbalance could easily lead to the prediction results completely biased towards the majority class, or the results may be very unstable. Therefore, we further investigated several data sampling methods to explore whether they could alleviate the data imbalance problem: (1) SMOTE [15] (2) RUS [16] (3) SMOTEENN [17] (4) AdaBoost [18] (5) GAN [19]. SMOTE is an over-sampling method, RUS is an under-sampling method, SMOTEENN is a comprehensive method combining over-sampling and under-sampling, and AdaBoost is an integrated learning method. GAN is a neural network method that constructs two networks, a generator and a discriminator. These two models compete with each other so that the generator generates instances closer to the ground truth, while the discriminator is getting stronger to identify the fake instances and ground truth.

TABLE VIII. THE PERFORMANCE OF DIFFERENT SAMPLING METHODS ON PREDICTION

Feature sets	metrics	AspectJ	Eclipse	JDT	SWT	Tomcat	avg
non-sample	F1	0.804	**0.812**	**0.802**	0.479	0.770	0.734
	ACC	**0.966**	**0.940**	**0.949**	0.920	0.987	0.952
SMOTE	F1	**0.878**	0.735	0.773	0.668	0.650	0.741
	ACC	0.964	0.866	0.912	0.854	0.940	0.907
SMOTE-ENN	F1	0.790	0.686	0.733	0.626	0.619	0.691
	ACC	0.927	0.815	0.877	0.802	0.918	0.868
Adaboost	F1	0.814	0.801	0.789	**0.732**	**0.827**	**0.793**
	ACC	**0.966**	0.938	0.948	**0.943**	**0.988**	**0.957**
GAN	F1	0.478	0.730	0.733	0.479	0.495	0.583
	ACC	0.917	0.925	0.937	0.920	0.980	0.936

TABLE IX. THE PERFORMANCE OF DIFFERENT SAMPLING METHODS ON SEVERITY

Feature sets	metrics	Eclipse	JDT	SWT	Tomcat	avg
non-sample	F1	0.848	0.859	0.883	0.840	0.858
	ACC	0.856	**0.872**	0.913	0.877	0.879
SMOTE	F1	0.841	0.848	0.872	0.819	0.845
	ACC	0.847	0.858	0.899	0.851	0.864
RUS	F1	0.864	**0.865**	0.893	0.842	0.866
	ACC	0.864	0.865	0.893	0.843	0.866
SMOTEENN	F1	0.838	0.842	0.842	0.797	0.830
	ACC	0.842	0.850	0.868	0.826	0.847
AdaBoost	F1	0.831	0.851	0.876	0.840	0.849
	ACC	0.840	0.864	0.909	0.877	0.872
GAN	F1	**0.876**	0.860	**0.896**	**0.894**	**0.882**
	ACC	**0.884**	0.869	**0.920**	**0.917**	**0.897**

Tables VIII and IX show the results. In priority prediction, AdaBoost outperforms the other sampling methods, while GAN is the worst. The advantages of SMOTE and SMOTEENN are not outstanding, as indicated by both enhancement and reduction. In severity prediction, GAN performs best, followed by RUS, which is slightly less accurate than AdaBoost. AdaBoost, SMOTE, and SMOTEENN showed an overall decrease in performance. The results suggest that over-sampling and comprehensive sampling may not be able to synthesize high-quality instances without sufficient data. Instead, RUS can avoid noise caused by synthesized instances when sufficient data is available. Note that GAN performs the worst in priority prediction but the best in severity prediction due to the fact that GAN requires a large amount of data for training. Therefore, we recommend using the GAN model for bug prediction when training data is sufficient.

V. DISCUSSION

In this section, we discuss the shortcomings of the research questions in our study.

Firstly, for *FSet-2*, we use CNN to extract the semantic features of source code files. As we know, CNN is a supervised model, the source code files associated with bug reports are considered as buggy. Therefore, additional clean files need to be fed to CNN for training along with the buggy files. We randomly select the remaining source files in the project as clean instances, which is consistent with the number of buggy instances. However, this may have potential noise, which leads to poor performance of *FSet-2*. First, not all the remaining source code files in the project are clean files. Second, since the mapping between bug reports and source code files is many-to-many, we can only determine the priority and severity of bug reports, not the source code files. Therefore, our treatment in this paper may lead to the learned features deviating from the actual predictions.

Secondly, for *FSet-3*, building a software network [5] based on the actual dependencies between code files may better represent their relationships. Since the projects in the datasets may across multiple versions, it is not possible to construct the corresponding software network accurately, so we use a co-occurrence network (COON) instead. Note that COON is actually a virtual network, in which the relationship between files is not completely real. Moreover, the stability of the network may change greatly as the dataset expands. This could also explain the unsatisfactory performance of the relational features.

Finally, the results show that the importance feature of the source code files is useful for *FSet-2*. In fact, our method involves only some of the source files, but the importance of these files is calculated from the perspective of the project, such as Key Class Identification [20]. Hence, there is still some deviation from the real situation.

VI. CONCLUSION

This study attempts to explore the impact of source code files feature sets on bug prioritization and severity prediction. Leveraging CNN and GNN to learn the semantic features of source code files and the relational features between these files. The experimental results show that the impact of learning feature sets is not as expected. In addition, five typical sampling methods are also introduced to analyze the impact on data imbalance. The results show that the GAN model synthesizes the highest quality instances with an adequate dataset, significantly improving the results on F-measure and Accuracy. Next is under-sampling, while over-sampling and comprehensive sampling do not perform well. In future work, in order to further validate the work of this paper accurately, we need to construct a dataset suitable for the method of this paper. The effects caused by other factors such as data quality should be excluded as much as possible, so that the source code file feature sets can be fully utilized.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Nos. 61832014, 61902114, 61977021), and the Key R&D Programs of Hubei Province (No. 2021BAA184, 2021BAA188).

REFERENCES

[1] M. Mihaylov, M. Roper. Predicting the Resolution Time and Priority of Bug Reports: A Deep Learning Approach, Ph.D. dissertation, Department of Computer and Information Sciences, University of Strathclyde, 2019.

[2] Tian Y, Lo D, Sun C. Drone: Predicting priority of reported bugs by multi-factor analysis[C]//2013 IEEE International Conference on Software Maintenance. IEEE, 2013: 200-209.

[3] Sharma M, Kumari M, Singh R K, et al. Multiattribute based machine learning models for severity prediction in cross project context[C]//International Conference on Computational Science and Its Applications. Springer, Cham, 2014: 227-241.

[4] Xiao Y, Keung J, Mi Q, et al. Improving bug localization with an enhanced convolutional neural network[C]//2017 24th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2017: 338-347.

[5] C. Zeng, C. Y. Zhou, S. K. Lv, P. He and J. Huang, "GCN2defect: Graph Convolutional Networks for SMOTETomek-based Software Defect Prediction," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), Wuhan, China, 2021, pp. 69-79.

[6] Sharma M, Kumari M, Singh R K, et al. Multiattribute based machine learning models for severity prediction in cross project context[C]//International Conference on Computational Science and Its Applications. Springer, Cham, 2014: 227-241.

[7] Umer Q, Liu H, Sultan Y. Emotion based automated priority prediction for bug reports[J]. IEEE Access, 2018, 6: 35743-35752.

[8] Ramay W Y, Umer Q, Yin X C, et al. Deep neural network-based severity prediction of bug reports[J]. IEEE Access, 2019, 7: 46846-46857.

[9] Bani-Salameh H, Sallam M. A Deep-Learning-Based Bug Priority Prediction Using RNN-LSTM Neural Networks[J]. E-Informatica Software Engineering Journal, 2021, 15(1) DOI:10.37190/e-Inf210102.

[10] McIntosh S, Adams B, Nagappan M, et al. Mining co-change information to understand when build changes are necessary[C]//2014 IEEE International Conference on Software Maintenance and Evolution. IEEE, 2014: 241-250.

[11] Zhang J, Wang X, Zhang H, et al. A novel neural source code representation based on abstract syntax tree[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 783-794.

[12] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[13] Wang S, Liu T, Nam J, et al. Deep semantic feature learning for software defect prediction[J]. IEEE Transactions on Software Engineering, 2018.

[14] Velickovic P, Fedus W, Hamilton W L, et al. Deep Graph Infomax[J]. ICLR (Poster), 2019, 2(3): 4.

[15] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research, 2002, 16: 321-357.

[16] Weiss G M. Foundations of imbalanced learning[J]. Imbalanced Learning: Foundations, Algorithms, and Applications, 2013: 13-41.

[17] Lamari M, Azizi N, Hammami N E, et al. SMOTE–ENN-Based Data Sampling and Improved Dynamic Ensemble Selection for Imbalanced Medical Data Classification[M]//Advances on Smart and Soft Computing. Springer, Singapore, 2021: 37-49.

[18] Ying C, Qi-Guang M, Jia-Chen L, et al. Advance and prospects of AdaBoost algorithm[J]. Acta Automatica Sinica, 2013, 39(6): 745-758.

[19] Sun Y, Jing X Y, Wu F, et al. Adversarial learning for cross-project semi-supervised defect prediction[J]. IEEE Access, 2020, 8: 32674-32687.

[20] Pan W, Song B, Li K, et al. Identifying key classes in object-oriented software using generalized k-core decomposition[J]. Future Generation Computer Systems, 2018, 81: 188-202.

Taint Trace Analysis For Java Web Applications

Yaju Li, Chenyi Zhang
Jinan University, Guangzhou, China

Qin Li
East China Normal University, Shanghai, China

***Abstract*—Taint analysis is concerned about whether a value in a program can be influenced, or tainted, by user input. Existing works on taint analysis focus on tracking the propagation of taint flows between variables in a program, and a security risk is reported whenever a *taint source* (user input) flows to a *taint sink* (resource that requires protection). However, a reported bug may have its taint source and taint sink located in different software components, which complicates the bug tracking and bug confirmation for developers. In this paper, we propose *Taint Trace Analysis* (TTA), which extends P/Taint, a context-sensitive Java taint analysis project, by making the taint information flow explicit. Thanks to the underlying Datalog semantics, we describe a way to extract traces of taint flows across program contexts and field accesses in the Doop framework. Different from existing works that produce only source-sink pairs, the output of TTA can be visualized as a set of traces which illustrate the inter-procedural taint propagation from taint sources to their corresponding sinks. As a consequence, TTA provides more useful information for developers and users after a vulnerability is reported. Our implementation is also efficient, and as shown in our experiment, it adds only a small run-time overhead on top of P/Taint for a range of analyses with different types of context-sensitivities applied.**

***Index Terms*—Taint analysis, Automatic trace generation, Program analysis, Java web application**

I. Introduction

Security vulnerabilities, which are software bugs exploitable by attackers, have been long standing challenges in software security and cyber security. As an example, Worm programs, such as the Microsoft SQL server Slammer [15], the Sun Telnet worm [26] and the Stuxnet worm [3], exploit software vulnerabilities in client or server programs and gain access to hundreds of thousands of new systems (including mobile phones [24]) within minutes. Of those notorious top security risks published by OWASP [17], Injection and Cross-Site Scripting (XSS) are usually triggered by crafted user input strings that are propagated through web application to reach their victims without censorship.

Taint analysis is an indispensable weapon in our combat against security vulnerabilities in system software, network software, and mobile applications. Existing approaches, including static taint analysis [18], [19] and dynamic taint analysis [16], [6], are implemented in tools such as TAJ [25], F4F [22] and Parfait [5] for tracking taint flows in web applications written in Java and JavaScript. However, provided that a complicated flow of taint may potentially pass through a number of program contexts via method calls and returns, the existing tools report only the taint sources with their corresponding taint sinks, which do not fully reveal all useful details that would guide a developer to identify or locate security vulnerabilities with ease.

Datalog has been used in program analysis since late 1990s [9]. In Datalog, program information is initially extracted into a group of base facts, from which advanced properties can be specified and subsequently computed in the style of logic programming. Such a reasoning pattern allows us to explicitly encode a trace of taint flow from the outputs of the existing points-to analysis and taint analysis facilities in the Doop framework [2]. Therefore, in this paper, we re-encode and extend the taint analysis constraints of P/Taint in order to support explicitly exporting a trace of information flow, taking advantage of Doop and its underlying Soufflé Datalog engine [11], and provide more useful information for developers when it is required to trace a reported vulnerability. In this paper, we have made the following contributions.

- We propose a method that explicitly exports a collection of taint traces associated with a given taint source-sink pair produced from a Java web application. The reported traces are context-sensitive, i.e., they contain all inter-procedural information of the complete taint flows.
- We have implemented our algorithm and conducted experiment on Securibench Micro [13]. The experiment results have confirmed that we only added a small overhead on top of the existing P/Taint [8] implementation in Doop.

The rest of the paper is organized as follows. In Section II, we illustrate how the proposed method works in a code snippet. Section III formally introduces the Taint Flow Analysis, with the implementation and experiment works presented in Section IV. The related works are discussed in Section V, and Section VI concludes our work.

II. A Motivating Example

The following example is simplified and adapted from Securibench Micro [13], as shown in Fig. 1. Note that in this example, a variable that points to the HttpServeletRequest object is set to receive data from client (user) input, and a reference to the HttpServeletResponse is used to write back to client. In particular, at line 3 of doGet() method, a user provided value is read from client and stored in variable name, which is then passed as a parameter in a call to method foo() at

*Corresponding email: chenyi_zhang@jnu.edu.cn.
This work is partially supported by the National Natural Science Foundation of China under Grant 62077028, Guangdong Natural Science Foundation under Grant 2019KTSCX010, Guangdong Basic and Applied Basic Research Foundation under Grant 2021A1515011873, Science and Technology Planning Project of Guangzhou under Grant 202102080307, and the Project of Guangxi Key Laboratory of Trusted Software (No. kx202007).
DOI reference number: 10.18293/SEKE2022-161

```
1  public class Example {
2    protected void doGet(HttpServletRequest req,
       HttpServletResponse resp) throws IOException {
3      String name = req.getParameter(''name'');
4      foo(name, ''abc'', req, resp);
5    }
6    ...
7    void foo(String str1, String str2,
       HttpServletRequest req, HttpServletResponse
       resp) throws IOException {
8      Data d = new Data();
9      d.value1 = str1;
10     d.value2 = str2;
11     PrintWriter writer = resp.getWriter();
12     writer.println(d.value1);    /* BAD */
13     writer.println(d.value2);    /* OK */
14   }
15 }
16 ...
17 class Data {
18   String value1;
19   String value2;
20 }
```

Fig. 1. A motivating example

line 4. P/Taint will report that there is a potential *Leak* at line 12, where a sink method println() is invoked with a tainted parameter d.value1. Since the tainted value is directly sent back to user, this could potentially form an XSS vulnerability, as the request information may consist of executable malicious code contained in a crafted link that a victim is fooled to click.

In order to trace a problem reported by a taint analysis and rule out false positives, a developer usually needs to manually inspect the source code and track the associated taint flows throughout the program. Such a procedure can be tedious and error-prone, as modern day web applications often consist of numerous software packages with included code written by third-party developers. Moreover, for object-oriented programming languages such as Java, a taint flow typically consists of a serial of inter-procedural method calls as well as field access of a value via alias references under different contexts, which further complicates the job of a code inspector. Taking a closer look at the reported *Leak* at line 12 in Fig. 1, the value loaded from d.value1 is previously stored and passed as a parameter at line 4 in a call to method foo, such that we have a tainted value passing through inter-procedural value-flow, stored in heap before being loaded back and forwarded to a sink method. For larger programs, a taint trace could be more complicated. In this case, Taint Trace Analysis (TTA) is able to automatically produce a corresponding taint trace (shown in Fig. 2) and visualize the flow of values, for a better comprehension and confirmation of reported security vulnerabilities.

The produced trace in Fig. 2 has six components, each written in the form of ⟨class : method(signature)/reference⟩. The first component is HttpServletRequest.getParameter(), a pre-defined *taint source*, with its value passed to the second component, a local variable name (line 3 in the code) of type string. The second component then passes the flow to

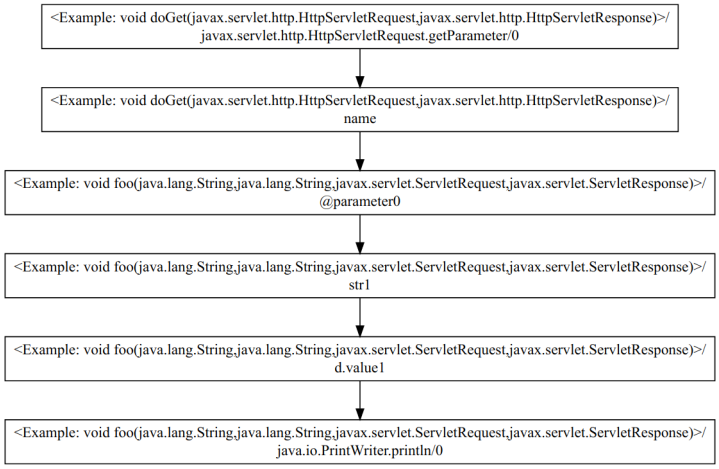


Fig. 2. A taint trace for the vulnerability reported for source code in Fig. 1.

the third component which is the first parameter of method foo (line 4 in the code), represented by @parameter0 in the trace. Eventually the flow goes to the last component println() which is a pre-defined *taint sink* (line 12 in the code).

III. Context Sensitive Taint Trace Analysis

We describe our algorithm on a simplified version of Java, with its syntax shown in Fig. 3.[1] A unique entry method such as main() is always assumed to appear in one of the classes in a well formed program. For each method, we assume that all local variables are defined at the beginning of the method, and there is always a dedicated local variable to be returned at the end. For an object allocation statement $x = \mathsf{new}^o\ A$, we assume that o uniquely identifies the allocation site. Given a program from the language, we define VAR for the set of variables, OBJ for the set of (abstract heap) objects, $\mathcal{C}$ for the set of classes, FLD for the set of fields, METHOD for the set of method, SIG for the set of method signatures, and INST the set of instructions for uniquely identifying call-sites.

We build our work on top of P/Taint [8] which is part of the program analysis framework Doop [2]. In order to generate facts for Datalog based reasoning, Doop makes use of the front end of Soot, a compiler framework for Java, for extraction of the program structural relations and basic flow relations, as shown in Table I. The extracted patterns are assumed to be self-explained on the right hand side of each relation of the table. For example, $\textsc{Alloc}(x, o)$ means there exists a statement $x = \mathsf{new}^o\ A$, which allocates object o to local variable x. Note that at the same time, $\textsc{HeapType}(o, A)$ is also added to the relation indicating that the class type of object o is A. Similarly, the existence of a relation $\textsc{Assn}(m, x, y)$ indicates that there is a statement $x = y$ in method m.

A. Taint Analysis with Points-to Analysis

First, we briefly review the P/Taint [8] project on which our work is based. At its foundation, Andersen-styled subset-based

[1]The actual implementation that handles the full Java language is more involved, and most of the implementation details are elided in this paper.

TABLE I
THE BASIC FLOW RELATIONS GENERATED FROM A PROGRAM.

ALLOC (v : VAR, o : OBJ)	// $v = \mathsf{new}^o\ A$
ASSN (m : METHOD, to : VAR, $from$: VAR)	// $to = from$ in method m
LOAD (to : VAR, $base$: VAR, f : FLD)	// $to = base.f$
STORE ($base$: VAR, f : FLD, $from$: VAR)	// $base.f = from$
VCALL ($base$: VAR, m : SIG, $inst$: INST)	// ... = $base.m(...)$ in instruction $inst$
FORMALARG ($meth$: METHOD, $i : \mathbb{N}$, v : VAR)	// v is the i-th formal arg of $meth$
ACTUALARG ($inst$: INST, $i : \mathbb{N}$, v : VAR)	// v is the i-th actual arg at callsite $inst$
FORMALRET ($meth$: METHOD, v : VAR)	// return v is a statement in $meth$
ACTUALRET ($inst$: INST, v : VAR)	// $v = base.m(...)$ in instruction $inst$
THISVAR ($meth$: METHOD, $this$: VAR)	// "$this$" as a special variable in $meth$
HEAPTYPE (o : OBJ, $A : \mathcal{C}$)	// object o is class A
LOOKUP ($A : \mathcal{C}$, m : SIG, $meth$: METHOD)	// through signature m, one can find the // corresponding method $meth$ in class A

$$
\begin{array}{lcl}
C & ::= & \mathsf{class}\ A()\ [\mathsf{extends}\ B]\ \{\overline{F};\ \overline{M}\} \\
F & ::= & A\ f \\
D & ::= & A\ z \\
M & ::= & m(\overline{x})\ \{\overline{D}; s; \mathsf{return}\ y\} \\
s & ::= & x{=}\mathsf{new}^o\ A \mid x{=}e \mid x.f{=}y \mid s; s \\
e & ::= & \mathsf{null} \mid x \mid x.f \mid x.m(\overline{y}) \\
prog & ::= & \overline{C}
\end{array}
$$

Fig. 3. Abstract syntax for the core language.

TABLE II
THE BASIC POINTS-TO RELATIONS

VARPT (v : VAR, o : OBJ)	// v points to object o
FLDPT (o_1 : OBJ, f : FLD, o_2 : OBJ)	// $o_1.f$ points to o_2

VARPT (v, o)	ALLOC (v, o).
VARPT (v_1, o)	ASSN (_, v_1, v_2), VARPT (v_2, o).
VARPT (v, o_1)	LOAD (v, $base$, f), VARPT ($base$, o_2), FLDPT (o_2, f, o_1).
FLDPT (o_1, f, o_2)	STORE (v_1, f, v_2), VARPT (v_1, o_1), VARPT (v_2, o_2).

points-to analysis [1] is applied for generating the points-to relations. For object-oriented languages such as Java, two relations are generated, VARPT, which assigns a set of objects to a variable (or reference), and FLDPT, which represents the relationship between (abstract) heap objects via fields. As shown in Table II, each relation given in the left hand side column is defined by conditions specified in the same row of the right hand side column. Taking VARPT, initially, v points to o if object o is allocated to v. The second rule extends the relation by adding $\text{VARPT}(v_1, o)$ if there is an assignment $v_1 = v_2$ and v_2 points to o (i.e., $\text{VARPT}(v_2, o)$ is already in the relation). Likewise, the last rule adds $\text{FLDPT}(o_1, f, o_2)$ if o_1 is pointed by variable v_1, o_2 is pointed by v_2, and $v_1.f = v_2$ is a (store) statement in the program.

Taint analysis is integrated with points-to analysis in P/-Taint [8], based on the fact that taint flow can be treated in the way similar to value flow through which an object is passed to a variable. We walk through the following reasoning patterns for taint analysis as shown in Table III. The first rule defines that if method m is a taint source and instruction i calls m with a value returned to variable to, then there is a taint flow to variable to, where TAINT is a constructor that introduces taint values in a new domain TAINTVALUE.[2] This definition plays the same role as the first rule (with $\text{ALLOC}(v, o)$) in Table II, since both rules are the base cases for their corresponding recursive relations, VARPT and TFLOW, respectively. The second rule in Table III resembles the second rule in Table II, by extending relations with assignments. Finally, the last rule in Table III defines the LEAK relation reporting that a taint value has reached a sink method, where $\text{SINK}(m, n)$ states that the n-th parameter of method m is a pre-defined *sink* location, and i is a call-site instruction that calls m with its n-th parameter v reachable from a taint source.

TABLE III
A FEW SAMPLE TAINT FLOW RULES

TFLOW (x : VAR, v : TAINTVALUE)	// taint value v flows to x
LEAK (v : TAINTVALUE, $inst$: INST)	// taint value v flows to // sink location $inst$

TFLOW (to, $value$)	SOURCE (m, $type$), CALLGRAPH (i, m), ACTUALRET (i, to), TAINT(i, $type$) = $value$.
TFLOW (v_1, $value$)	ASSN (_, v_1, v_2), TFLOW (v_2, $value$).
LEAK ($value$, i)	CALLGRAPH (i, m), SINK (m, n), ACTUALARG (i, n, v), TFLOW (v, $value$).

Doop also supports a range of context-sensitivity options to strengthen its points-to and taint flow computation, such as call-site-sensitivity [20], object-sensitivity [14], [12], and type-sensitivity [21]. The enhanced VARPT relation has four components.

$$\text{VARPT}(c_1 : \text{CTX},\ v : \text{VAR},\ c_2 : \text{HCTX},\ o : \text{OBJ})$$

where c_1 is the context for variable v and c_2 is the context for object o. The variable context domain CTX and the heap context domain HCTX may be defined as distinct sets. Similarly, we have the following definition for the field points-to relation.

$$\text{FLDPT}\ (c_1 : \text{HCTX},\ o_1 : \text{OBJ},\ f : \text{FLD},\ c_2 : \text{HCTX},\ o_2 : \text{OBJ})$$

[2]Here the $\text{CALLGRAPH}(i, m)$ relation is taken as the context insensitive version of the CALLGRAPH relation in Table IV. Intuitively, we retrieve the class type of the object pointed-to by the receiver variable at the call-site i, from which we match the signature of method m.

TABLE IV
SAMPLE RULES FOR k-OBJECT-SENSITIVE CONTEXT EXTENSION

REACHABLE (c : CTX, m : METHOD)
// method m is reachable with context c
CALLGRAPH (c_1 : CTX, i : INST, c_2, m : METHOD)
// method m is called at call-site i

REACHABLE ([ ], main).	
REACHABLE (c_2, m), CALLGRAPH (c_1, i, c_2, m)	REACHABLE (c_1, m_0), CONTAIN(m_0, i), VCALL (x, m', i), NEWCTX (c_1, o) = c_2, VARPT (c_1, x, c_3, o), HEAPTYPE (o, A), LOOKUP (A, m', m).

TABLE V
RELATIONS FOR ONE-STEP VALUE FLOW

// variable $from$ in context c_2 flows to variable to in context c_1
VFLOW (c_1: CTX, to : VAR, c_2: CTX, $from$: VAR)
// variable v_1 in context c_1 and v_2 in context c_2 are in the "may alias" relation
ALIAS (c_1: CTX, v_1 : VAR, c_2: CTX, v_2 : VAR)

ALIAS (c_1, v_1, c_2, v_2)	VARPT (c_1, v_1, c_3, o), VARPT (c_2, v_2, c_3, o).
VFLOW (c, to, c, $from$)	REACHABLE (c, m), ASSN (m, to, $from$).
VFLOW (c_1, x_1, c_2, x_2)	REACHABLE (c_1, m_1), LOAD (m_1, x_1, v_1, f), REACHABLE (c_2, m_2), STORE (m_2, v_2, f, x_2), ALIAS (c_1, v_1, c_2, v_2).
VFLOW (c_1, x_1, c_2, x_2)	CALLGRAPH (c_2, i, c_1, m), FORMALARG (m, n, x_1), ACTUALARG (i, n, x_2).
VFLOW (c_1, x_1, c_2, x_2)	CALLGRAPH (c_1, i, c_2, m), FORMALRET (m, x_2), ACTUALRET (i, x_1).
VFLOW (c_1, x_1, c_2, x_2)	VCALL (x_2, _, i), CALLGRAPH (c_2, i, c_1, m), THISVAR (m, x_1).

In Doop, new contexts are always introduced at a call-site, where variables and objects defined within the callee method acquire a new context by combining the caller context and the information associated with the caller instruction. As an example, for k-call-site sensitivity, a new context is created by appending the current call-site to the caller context (which is a list of call-sites), and deleting the oldest component (usually the first call-site in the current list) if the newly-formed context exceeds the predefined limit k. New contexts in a k-object-sensitive analysis are constructed in a similar way, with details given as follows.

We present an example of constructing new contexts at a call-site for a k-object-sensitivity analysis in Table IV, where two relations REACHABLE and CALLGRAPH are recursively defined. REACHABLE(c, m) denotes that method m is reachable with context c, and main, the entry method, is always reachable with the (pre-defined) empty context "[ ]". CALLGRAPH is used to extend a caller context to a callee context at a call-site. As the rule explains, given a call instruction i of the form "$\ldots = x.m'()$", if the context of the enclosing method (say m_0) of call-site i is c_1, and the matched method for the call-site is m (via LOOKUP), then m is reachable in context c_2. At the same time, we establish a CALLGRAPH relation from i in context c_1 to m in context c_2. Note that the semantics of context constructor NEWCTX depends on which context-sensitivity is applied. Since we assume k-object-sensitivity (i.e., a context is a list of heap objects), if the length of c_1 is less than k, then $c_2 = c_1 \circ o$, where "$\circ$" appends o to c_1; otherwise, c_2 is the list of removing the first object from $c_1 \circ o$.

B. Generating the One-Step Flow Relation

P/Taint only reports source-sink pairs in the form of LEAK$(c_1, value, c_2, invo)$. With this much information, it is often difficult for developers to find out how the taint source "$value$" in context c_1 is propagated to the sink location "$invo$" in context c_2, as the actual value flow may have passed through a number of inter-procedural calls and returns, as well as loading a value previously stored in the heap. In this section, we define new relations that help to make the flow of taint explicit. We extend the P/Taint system with the logic shown in Table V. To simplify the presentation, we assume k-object-sensitivity is applied, and the NEWCTX(o, c) constructor in Table V produces a new context by appending object o to context c and then removing the first component of c if the length of c is already k (so that after appending o, the new context does not have length exceeding k).

We define two new relations in this extension. ALIAS(c_1, v_1, c_2, v_2) can be straightforwardly derived from the available context-sensitive points-to analysis, in the sense that if two variables (probably in different contexts) may-points-to the same object, then they are in ALIAS relation. The construction of VFLOW relation is a bit more involved, as explained in the following three cases.

1) Variable $from$ flows to to in context c, if the enclosing method m of "$to = from$" is reachable in context c;
2) Given (c_1, v_1) and (c_2, v_2) in alias relation with both enclosing methods m_1 and m_2 reachable in context c_1 and c_2, respectively, and if there is a LOAD of v_1 via field f to variable x_1 and a STORE into v_2 via f from x_2, then this "load/store" can be paired to serve as a bridge which directs a value flow from x_2 to x_1;
3) The last three rules describe the inter-procedural value flow passing through a parameter, the return value, and this reference, respectively. Taking the last rule as an example, given a call-site from which an inter-procedural call (represented by the CALLGRAPH relation) can be established at call-site i from context c_2 to c_1 where the callee method is identified as m, we construct a one-step flow from the receiver x_2 (in context c_2) to this of m in context c_1.

C. Generating Traces for Taint Analysis

Formally, our analysis exports finite traces of the form $[w_0, w_1, w_2 \ldots w_k]$, where $w_i = (c_i, x_i)$ with value x_i in context c_i for all $1 \leq i \leq k-1$. A valid trace is required to satisfy the following three conditions.

1) Variable x_1 receives a return value from the pre-defined taint source v_0 in context $c_1 \in$ CTX.
2) Variable x_i in context c_i performs a one-step flow to variable x_{i+1} in context c_{i+1} for all $1 \leq i < k-1$.
3) Variable x_{k-1} passes value to a taint sink w_k.

TABLE VI
SAMPLE RULES FOR TAINT TRACE GENERATION

.type list = [next:list, x:ctype]
TTRACE ($trace$: list) // a taint trace is a list of values
INTTRACE ($trace$: list) // an intermediate trace

INTTRACE ([[nil, w_1], w_2])	CALLGRAPH (c, i, _, m), ACTUALRET(i, x), $w_2 = (c, x)$, $w_1 = (c, i)$, SOURCE (m, $type$).
INTTRACE ([t, w])	t = [t_1, w_1], $w_1 = (c_1, y)$, INTTRACE (t), $w = (c, x)$, VFLOW(c, x, c_1, y).
TTRACE ([t, w])	t = [t_1, w_1], $w_1 = (c, x)$, INTTRACE (t), SINK (m, n), CALLGRAPH (c, i, _ m), ACTUALARG (i, n, x), $w = (c, i)$.

The construction of each trace relies on the recursively typed records supported by the Datalog engine Soufflé [11]. For example, in order to encode a list of values, one needs to define a type list by [list, ctype] where ctype is the (component) type for the values in the list, and the recursion is terminated by a special predefined constant list nil. We present the definition of a taint trace (TTRACE) in Table VI. In this formulation, an intermediate trace (INTTRACE) encodes a list of component starting with a pre-defined source location (but it has not reached a sink location). The algorithm then tries to extend the list by looking for elements that are reachable from the last element in the existing list, following the VFLOW relation. Once a INTTRACE reaches a pre-defined taint sink, the search terminates with a complete TTRACE ready for output.

IV. IMPLEMENTATION AND EXPERIMENT

We have implemented our algorithm in Doop which extracts Datalog facts from the Shimple code which is an SSA-based Intermediate Representation (IR) in Soot. The experiment is conducted on a laptop equipped with an Intel Core® i5-8250U CPU@1.60GHz*8 and 16GB RAM, running UnionTech OS GNU/Linux 5.4.50-amd64-desktop. The software environment includes Doop (version 4.24.2), soufflé (version 2.0.2), graphviz (version 2.49.3), python (version 3.8.0) and OpenJDK (version 1.8.0). Our implementation is publicly available at https://github.com/lyj18688610256/LDoop.[3]

To measure the efficiency of our implementation, we carry out an experiment on Securibench Micro, an open source benchmark suite with 140 pairs of known source-to-sink flows located in a range of small web applications [13], from which we evaluate the precision, recall and efficiency of TTA. Different from the other methods that merely output ⟨source, sink⟩ pairs, we produce a set of graphically represented traces (with a context at each node of the trace) similar to what is shown in Fig. 2. We then start to manually check the traces in order to verify whether there are *false positives*. Fortunately, this procedure is not very time consuming, thanks to all taint flows being made explicit. We observe that sometimes our analysis produces a trace that contains a circle, i.e., $v_i = v_j$ with $0 < i < j < k$ where k is the length of the trace. One strategy that we currently use is to set up a search limit for traces, which rules out most of the redundant trace being reported. However, it should be noticed that some taint flows will be beyond our reach and becomes *false negatives* if the depth is too short. Currently, we are using an empirical limit for the Securibench Micro suite for an optimal result.

[3]A substantial amount of effort has been made to ensure that TTA aligns with the expected P/Taint analysis results. Due to recent updates in Doop, it is currently infeasible to directly reproduce the results as given in [8].

TABLE VII
ACCURACY OF TTA ON SECURIBENCH MICRO (2-OBJECT-SENSITIVE+HEAP)

Suite	TP	FP	FN	Precision	Recall	F-score
Total	131	20	9	87%	94%	90%
aliasing	11	0	1	100%	92%	96%
arrays	10	3	1	77%	91%	83%
basic	61	0	0	100%	100%	100%
collections	16	3	0	84%	100%	91%
datastructures	6	0	0	100%	100%	100%
factories	3	0	0	100%	100%	100%
inter	11	6	5	65%	69%	67%
pred	3	5	0	38%	100%	55%
reflection	4	0	0	100%	100%	100%
sanitizers	2	0	2	100%	50%	67%
session	3	1	0	75%	100%	86%
strong_updates	1	2	0	33%	100%	50%

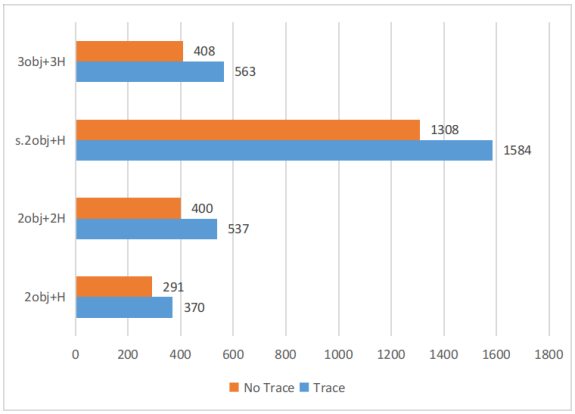


Fig. 4. Run-time (in seconds) with and without trace generation for different context-sensitivity options. For example, $3obj + 3H$ means that (lists of) at most 3 objects are used to represent method/variable contexts in CTX and also at most 3 objects are used to represent heap contexts in HCTX.

Accuracy of TTA: The accuracy results of TTA with the sensitive 2-object-1-heap analysis (i.e., $s.2obj + H$) on the benchmark suite are shown in Table VII, where **TP**, **FP** and **FN** represent *true positive*, *false positive* and *false negative*, respectively. From the table, one may find that for our implementation, the worst performed portion of the benchmark suite is the strong_updates folder (33% precision), which is due to that value flows for strong update usually require a *must* PTA, while the current TTA is based on *may* PTA. Nevertheless, over all, we still achieves 87% precision and 94% recall, with a 90% F_1 score. This effectively shows that TTP produces an acceptable result for the current benchmark suite.

Performance of TTA: We also check the run-time cost of our TTA implementation compared to the original P/Taint algorithm [8], with results shown in Fig. 4. When running the benchmark suite, for each context-sensitivity option, we ana-

lyze all programs in Securibench Micro simultaneously. From the table, one may find that our trace generation algorithm only adds a small amount of run-time overhead (less than 38%) to the original algorithm. Although the run-time cost of our implementation increases with larger contexts, for $2obj + H$ and $3obj + 3H$, the increased time cost becomes negligible. One possible explanation is that although $3obj + 3H$ analysis needs more time to compute variable contexts, it potentially produces less false positives. Therefore, the amount of time required for computing the traces becomes less.

V. Related Work

Perhaps the earliest taint analysis was implemented as the "taint mode" in the Perl language, which tracks taint flows via data-dependencies at runtime, i.e., taintness flows from the right hand side of each assignment to the left hand side, and reports an error if the arguments of a system call is tainted [4]. In general, a dynamic taint analysis tracks taint information at program execution, implemented with various methodologies such as binary instrumentation [16], [6], hardware-based taint tracking [23], and compiler-based taint tracking [27]. Static taint analysis heavily relies on underlying tools and representations of a program, including control flow graph [19], program dependence graph [10], program slicing [18], and type system [7]. In general, dynamic taint analysis provides security guarantee at runtime, while static analysis achieve better coverage at the cost of false positives. Our work is in the category of static taint analysis, with powerful context-sensitivity options from the underlying Doop framework and partial flow-sensitivity from the Shimple IR of Soot. Therefore, we are able to report potential vulnerabilities with relatively low false positive rate (c.f. Section IV). Moreover, our focus is on producing explicit taint traces for software developers.

Our work is an extension on P/Taint [8], a unified points-to analysis and taint analysis in the Doop framework [2] equipped with a range of context-sensitivity options. Given that P/Taint only reports source-sink pairs which are hard to comprehend by human, we explore ways of trace generation from the explicit flow relation in Type Flow Analysis [28], which express a flow relation by joining intraprocedural flow, inter-procedural flow, and pairing of load and store. Such a one-step relation can be connected with a recursive encoding technique supported in the Datalog engine Soufflé [11].

VI. Conclusion and Future Work

We have introduced an extension for context-sensitive taint analysis for Java (P/Taint), called Taint Trace Analysis (TTA), which produces a set of taint traces with context information included. Little run-time overhead on top of P/Taint has been shown for our implementation. In contrast to most existing taint analysis works, the produced taint traces from TTA may provide more useful information for the detection and tracking of security vulnerabilities in Java web applications. In the future, we will extend the existing work to analyze Android apps, and we also plan to further optimize our algorithm to achieve higher precision and a quicker visualization procedure.

References

[1] L. O. Andersen. *Program analysis and specialization for the C programming language*. PhD thesis, DIKU, University of Copenhagen, May 1994.

[2] M. Bravenboer and Y. Smaragdakis. Strictly declarative specification of sophisticated points-to analyses. In *OOPSLA'09*, page 243–262, 2009. https://bitbucket.org/yanniss/doop/.

[3] T. Chen and S. Abu-Nimeh. Lessons from stuxnet. *IEEE Computer*, 44(4):91–93, 2011.

[4] T. Christiansen. Perl security. https://perldoc.perl.org/. Taint module available November 1997.

[5] C. Cifuentes, N. Keynes, L., N. Hawes, and M. Valdiviezo. Transitioning parfait into a development tool. *IEEE Security and Privacy*, 10(3):16–23, 2012.

[6] J. Clause, W. Li, and A. Orso. Dytan: A generic dynamic taint analysis framework. In *Proceedings of the 2007 International Symposium on Software Testing*, pages 196—-206, 2007.

[7] J. S. Foster, T. Terauchi, and A. Aiken. Flow-sensitive type qualifiers. In *PLDI'02*, pages 1—-12, 2002.

[8] N. Grech and Y. Smaragdakis. P/Taint: unified points-to and taint analysis. In *OOPSLA'17*, pages 102:1–102:26, 2017.

[9] B. Greenman. Datalog for static analysis. History of programming language seminar (2017) https://github.com/nuprl/hopl-s2017.

[10] C. Hammer, J. Krinke, and G. Snelting. Information flow control for java based on path conditions in dependence graphs. In *IEEE International Symposium on Secure Software Engineering*, 2006.

[11] H. Jordan, B. Scholz, and P. Subotić. Soufflé: On synthesis of program analyzers. In *CAV'16*, pages 422–430, 2016.

[12] G. Kastrinis and Y. Smaragdakis. Hybrid context-sensitivity for points-to analysis. In *PLDI'13*, pages 423–434, 2013.

[13] B. Livshits. *Improving Software Security with Precise Static and Runtime Analysis*. PhD thesis, Stanford University, 2006.

[14] A. Milanova, A. Rountev, and B. G. Ryder. Parameterized object sensitivity for points-to analysis for Java. *ACM Transations on Software Engineering and Methodology*, 14(1):1–41, 2005.

[15] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security and Privacy*, 1(4):33–39, 2003.

[16] J. Newsome and D. Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *NDSS'05*, 2005.

[17] OWASP. Top 10 web application security risks. https://owasp.org/www-project-top-ten/. Accessed: 2021-11-23.

[18] M. Pistoia, L. Koved R. J. Flynn, and V. C. Sreedhar. Interprocedural analysis for privileged code placement and tainted variable detection. In *ECOOP'05*, pages 362—-386, 2005.

[19] B. Scholz, C. Zhang, and C. Cifuentes. User-input dependence analysis via graph reachability. In *SCAM'08*, pages 25—-34, 2008.

[20] O. G. Shivers. *Control-flow Analysis of Higher-order Languages or Taming Lambda*. PhD thesis, Carnegie Mellon University, 1991. UMI Order No. GAX91-26964.

[21] Y. Smaragdakis, M. Bravenboer, and O. Lhoták. Pick your contexts well: understanding object-sensitivity. In *POPL'11*, pages 17–30, 2011.

[22] M. Sridharan, S. Artzi, M. Pistoia, S. Guarnieri, O. Tripp, and R. Berg. F4F: taint analysis of framework-based web applications. In *OOPSLA 2011*, pages 1053–1068. ACM, 2011.

[23] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas. Secure program execution via dynamic information flow tracking. In *ASPLOS'04*, page 85–96, 2004.

[24] Symantec. Top 10 web application security risks. Internet Security Threat Report, Vol. 21., April 2016.

[25] O. Tripp, M. Pistoia, S. J. Fink, M. Sridharan, and O. Weisman. TAJ: effective taint analysis of web applications. In *PLDI'09*, pages 87–97. ACM, 2009.

[26] US-CERT. Vulnerability note vu#881872, sun solaris telnet authentication bypass vulnerability. http://www.kb.cert.org/vuls/id/881872. Accessed: 2021-11-23.

[27] W. Xu, S. Bhatkar, and R. Sekar. Taint-enhanced policy enforcement: A practical approach to defeat a wide range of attacks. In *Proceedings of the 15th USENIX Security Symposium*, page 121–136, 2006.

[28] X. Zhuo and C. Zhang. TFA: an efficient and precise virtual method call resolution for Java. *Formal Aspects of Computing*, 32:395—416, 2020.

Impact of Combining Syntactic and Semantic Similarities on Patch Prioritization while using the Insertion Mutation Operators

{Mohammad Raihan Ullah, Nazia Sultana Chowdhury, Fazle Mohammed Tawsif}
Institute of Information and Communication Technology
Shahjalal University of Science and Technology
Sylhet, Bangladesh

***Abstract*—Patch prioritization ranks candidate patches based on their likelihood of being correct. The fixing ingredients that are more likely to be the fix for a bug, share a high contextual similarity. A recent study shows that combining both syntactic and semantic similarity for capturing the contextual similarity, can do better in prioritizing patches. In this study, we evaluate the impact of combining the syntactic and semantic features on patch prioritization using the Insertion mutation operators. This study inspects the result of different combinations of syntactic and semantic features on patch prioritization. As a pilot study, the approach uses genealogical similarity to measure the semantic similarity and normalized longest common subsequence, normalized edit distance, cosine similarity, and Jaccard similarity index to capture the syntactic similarity. It also considers Anti-Pattern to filter out the incorrect plausible patches. The combination of both syntactic and semantic similarity can reduce the search space to a great extent. Also, the approach generates fixes for the bugs before the incorrect plausible one. We evaluate the techniques on the IntroClassJava benchmark using Insertion mutation operators and successfully generate fixes for 6 bugs before the incorrect plausible one. So, considering the previous study, the approach of combining syntactic and semantic similarity can able to solve a total number of 25 bugs from the benchmark, and to the best of our knowledge, it is the highest number of bug solved than any other approach. The correctness of the generated fixes are further checked using the publicly available results of CapGen and thus for the generated fixes, the approach achieves a precision of 100%.**

***Index Terms*—patch prioritization, semantic similarity, syntactic similarity, automated program repair, mutation operation, contextual similarity**

I. Introduction

Automated Program Repair is the automatic repair of software bugs without the intervention of a human programmer. Automatic debugging and repairing of the software defects can make a programmer's life a lot easier.

Manual Debugging is a very difficult, time-consuming, and painful task. Study shows that the global cost of debugging is 312 billion dollars annually and software developers spend 50 percent of their time fixing bugs [1].

Automated Program Repair (APR) reduces the bug-fixing effort by automatically generating patches that satisfy a specification e.g. Test cases [2]. A patch is a modification applied to a program for fixing a bug. A typical APR approach takes a faulty program and a set of test cases as input (where the program fails at least one test case) and produces a patch to repair the fault (the patched program at least passes all the tests). This type of technique is called the generate-and-validate technique. Patches that are applied to a program could exist in the faulty program itself as well as in a non-local program. The study founds that up to 69% of the repaired code fragments (in the form of code lines) can be obtained from both the local program and non-local programs [3], [4].

Throughout the years, several Automated Program Repair (APR) techniques have been introduced. Such as GenProg, HDRepair, CapGen, ssFix, SimFix, Elixir, etc [7], [8], [9], [10], [11], [12], [13], [14]. Many of these techniques e.g. SimFix, ssFix, CapGen, Elixir, etc [9], [10], [12], [15], [16] are known as search-based APR. So, APR is often described as a search problem in a patch candidate space. In a candidate space, there are the fixing code fragments or patches that can be a possible fix for that particular bug. Testing the fixing codes earlier that are more likely to be the fix for the bug can reduce bug fixing time. Sorting candidate patches based on their probability of being the correct fix for the bug, is called patch prioritization.

The faulty program and the code fragment where the fix resides share a high contextual similarity. To capture these similarity, some method [18] [19] [20] uses Syntactic features and some method [22] uses Semantic features. A recent study [5] shows that combining Syntactic and Semantic features in patch prioritization can reduce bug fixing time and achieve higher precision. So in this study, we use the combination of Syntactic and Semantic features in Patch Prioritization and observe the impacts while working with *Insertion* as the mutation operation. We also evaluate some different ways of combining those feature scores to calculate the final score and analyze their impact on Patch Prioritization. We evaluate the techniques on IntroClassjava benchmark [23] which contains 297 real-life bugs. We were successfully able to solve six bugs with a precision of 100%.

II. Related Work

Testing the candidate patches early which, are more likely to be the fix can help to reduce the bug fixing time and give some

DOI reference number: 10.18293/SEKE2022-047

treatment to the search space explosion problem of search-based Automated Program Repair.

The most representative of the search-based APR category is GenProg [17], which searches for correct patches via genetic programming algorithm. GenProg constrains the genetic operations of mutation and crossover to operate only on the region of the program that is relevant to the error. GenProg repairs bugs with the help of the following components: Mutation Space, Crossover, Evaluation Function. GenProg presents a program as an Abstract Syntax Tree (AST) and modifies the faulty node using the mutation operations such as insertion, deletion, and Replacement operators. AST of the program is modified using mutation from the mutation space. As a result of the modification, variants of the faulty program are generated then they are passed to the evaluation function to see if those variants pass the test cases.

Elixir [18] generates patches based on generate-and-validate techniques. For a given bug, Elixir takes as input a buggy program, a test suite, and optionally a bug report, then produces a patch that passes all the test cases. Elixir works in the following ways: Bug Localization, Generating Candidate Patches, Ranking and Selection of Candidate Patches, Validating Selected Candidate Patches. Elixir uses four syntactical features to rank the patches. They are Distance score between faulty and fixing ingredients, Contextual Similarity, Frequency in the context, Bug Report Similarity. The approach introduces 8 templates and validates only the top 50 patches generated from each template.

SimFix [19] defines a method to obtain a search space from existing patches based on an abstract space definition on AST types. The approach in SimFix consists of - an offline mining stage, an online repairing stage. SimFix uses the Ochiai algorithm to localize fault which produces an ordered list of suspicious statements. It extracts a set of fixing ingredients using syntactic features, such as Structure Similarity, Variable Name Similarity, Method Name Similarity. Then it matches variables in the faulty and fixes ingredients and builds a variable mapping table for code adaptation. It calculates the difference between faulty and fixing ingredients and extracts concrete modification. Finally, it generates a list of patches and validates them using the test suite. These three features obtain low precisions 63.41%, 33.33%, and 60.7%, respectively.

ssFix [20] uses the syntactic features to calculate the similarity between faulty code and fixing ingredients. It uses the TF-IDF model to calculate the similarity score. They evaluated their technique on 357 bugs in the Defect4j [21] bug dataset and successfully solved 20 bugs from that dataset.

CapGen [22] uses semantic features to prioritize patches. It works on the fine-grained level to extract fixing ingredients. They selected 30 augmented mutation operators to generate operation space. For example: insert Simple Name under Method Invocation or delete Expression Statement under Method Declaration etc. CapGen uses three models to calculate the similarity score between faulty nodes and fixing ingredients. They are Genealogy Context, Accessed Variable, Semantic Dependency. CapGen combines the three model scores to calculate the final score. Based on that score, it prioritizes the patches and tests the patches which are more likely to be the fix for the bug. This approach achieves a precision of 84%.

A study [17] showed that combining syntactic and semantic features for patch prioritization can work better than using syntactic and semantic features alone. In the study, they took expression level bug and used replacement mutation operation for patch generation. To calculate the similarity between faulty code and fixing ingredients, it used Genealogical & variable similarity as semantic metrics and Normalized Longest Common Subsequence (Normalized LCS) as syntactic metrics. The approach was evaluated using 22 bugs from the IntroClassJava benchmark [23], fixed by CapGen applying replacement mutation [22]. It repaired all 22 bugs and achieved a precision of 100%.

III. Methodology

A recent study [5] shows that combining both syntactic and semantic similarity can improve patch prioritization. It works at *Expression* level and uses *Replacement* mutation operators to generate the patches. In this research, we evaluate the technique further to observe its impact on patch prioritization while using the *Insertion* as the mutation operation. The approach works at the *Expression* and *Expression-Statement* level as the *Insertion* mutation operation works better at these level [22]. It manipulates the source codes representing them in Abstract Syntax Tree (AST). Figure 1 shows some sample bug fixes using *Insertion* Mutation Operation. In figure 1a, the bug is fixed by inserting a *Simple Name* under the faulty *Method Invocation*. In figure 1c, the bug is fixed inserting a *Method Invocation* under the *If-Statement*.

The approach in this paper takes a source program and a test suite (positive and negative test cases) as input and generates some repaired variants. For generating patches, it uses *Insertion* mutation operation using *Expression* and *Expression-Statement* level code for fixing elements. It uses Cosine Similarity, Jaccard Similarity Index, Normalized Edit Distance, Normalized LCS as a syntactic feature, and Genealogical similarity and Anti-Pattern as a semantic feature. The repaired variants are thus tested with the test suite and the correct patches for the bug are validated. The approach works in the four following steps. The steps are quite similar to the approach of the previous study [5].

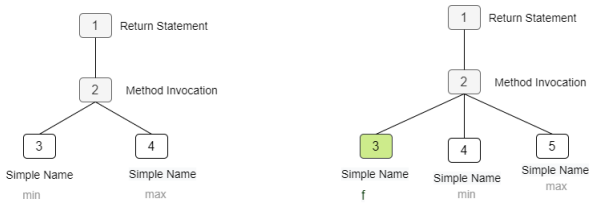
A. Fault Localization

This step identifies the faulty code using Ochiai [26] algorithm, a spectrum-based fault localization technique. It takes help from the GZoltar [25] tool for this purpose. GZoltar takes a source code and a test suite (including positive and negative test cases) and generates a list of the suspicious statements. For *Insertion* mutation operation, the approach identifies the faulty AST node of *Expression* and *Expression Statement*. It outputs a line number-wise suspicious value (probability of being a faulty node) of a program. If a node spans across multiple

```
//faulty code
72: - return solve (min, max);

// fixing ingredient
72: - return solve (f, min, max);
```

(a) A fixed bug (Math 70 from Defects4J) using *Insertion* mutation operation



(b) AST difference of the fixed bug 1a

```
421: if (escapingOn && c[start] == QUOTE)  // Buggy statement
422: + next(pos); // Fixed statement
423: return appendTo == null ? null : appendTo.append(c);

// Fixing ingredient for the bug
170: if (c[pos.getIndex()] == START) {
171: format = parse(pattern, next(pos));
```

(c) A Fixed Bug (Lang 43 in Defects4J) Using the *Insertion* Mutation Operation and Fix Extracted From the Same File

Fig. 1: Sample Bug Fixes using Insertion Mutation Operation

lines, it is assigned the average of the suspicious values of those lines.

B. Patch Generation

In this step, the approach generates patches inserting fixing ingredients under the faulty node. For faulty nodes, it takes *Expression* and *Expression-Statement* level bug. Ambiguity arises while inserting a node under a block statement. This ambiguity can be addressed as *fix location selection* problem. To avoid ambiguity for fix location, it further splits the insertion operation into two more categories:

- Insert Before: Insert a subtree before node *N*.
- Insert After : Insert a subtree after node *N*.

The operation space induced by *Insertion* mutation operator is huge. To reduce this space, we use the top 10 augmented insertion mutation operations from *CapGen* [22].

C. Patch Prioritization

Several patches are produced at the patch generation level. To minimize the bug fixing time and cost, patches are prioritized. It also helps to maximize the precision of patches. For prioritizing patches, the genealogical similarity between is used to measure semantic similarity, and normalized Longest Common Subsequence (LCS) and Anti - Pattern [6] and Cosine Similarity is used to measure syntactic similarity.

- **Genealogical Similarity:** The genealogical structure of an AST node shows a node is frequently used under and together with which types of code elements by checking its ancestor & sibling node [22]. Traversal is done from the node to its ancestor until a method declaration is found. And nodes of category Statement and Expression within the same Block are extracted for sibling nodes [22]. The Genealogical Similarity is measured as

$$gen\_s(f_n, f_e) = \frac{\sum_{t \epsilon k} \min\left(f_n(t), f_e(t)\right)}{\sum_{t \epsilon k} f_n(t)} \quad (1)$$

Here, f_n and f_e are frequencies of different node types for faulty node and fixing ingredient respectively. K indicates a set of all distinct AST node types captured by f_n

- **Normalized LCS:** LCS measures the Longest Common Subsequences between two sets of sequence. To measure the LCS, we take string representation of faulty node (f_n) and fixing ingredient(f_e). Then the value of the LCS is normalized to get the value in the interval [0,1].

$$n\_lcs((f_n, f_e)) = \frac{LCS(f_n, f_e)}{\max\left(f_n, f_e\right)} \quad (2)$$

- **Cosine Similarity:** The cosine similarity of two ASTs representative vectors expresses the similarity between them. The cosine similarity measure uses two finite-dimensional vectors of the same dimension in which each vector represents a document. Given two string f_n (string representation of faulty node) and f_e (string representation of fixing ingredient), we construct the collection between the two elements. The collection of terms denotes C = $\{c_1, c_2, ..., c_n\}$ with $c_i \in f_n \mid c_i \in f_e$ and each c_i is distinct. The strings are then represented in an n-dimensional vectors $\vec{V}_{f_n}$ and $\vec{V}_{f_e}$.

$$cos\_s(f_n, f_e) = \frac{\vec{V}_{f_n} . \vec{V}_{f_e}}{(|\vec{V}_{f_n}| * |\vec{V}_{f_e}|)} \quad (3)$$

- **Normalized Edit Distance**: Edit distance refers to the minimum number of edit operations needed to convert from one string to another string. For measuring the score, we use an alternative version of Levenshtein distance where each Insertion or Deletion cost is 1 and Substitution cost is 2. The score is then normalized to get the score between the interval [0,1]. Finally, the score is subtracted from 1 as the approach in this paper only considers the similarities between the faulty nodes and fixing ingredients.

$$n\_ed(f_n, f_e) = 1 - \frac{Edit\_Distance(f_n, f_e)}{\max\left(f_n, f_e\right)} \quad (4)$$

Here, f_n and f_e is the string representation of faulty node and fixing ingredient.

- **Jaccard Similarity Index:** The technique computes Jaccard distance between string representation of faulty node (f_n) and fixing node (f_e). The input string is converted into a set of N-grams. In Comparing with Cosine similarity, Jaccard Similarity Index takes the number of common attributes is divided by the number of attributes that exist

in at least one of the two objects. In comparison, Cosine similarity divides the number of the common attributes by the dot product of two objects represented in vectors.

$$jac\_s(f_n, f_e) = \frac{|f_n \cap f_e|}{|f_n \cup f_e|} \quad (5)$$

Here, f_n and f_e is the string representation of faulty node and fixing ingredient.

- **Anti-Pattern:** Anti-patterns capture disallowed modifications to the buggy program [24]. If such a modification passes all tests in the given test-suite, they are not counted as repairs [24]. Because those modifications may later introduce some code smell in the program. We filter out such modifications as part of patch prioritization. We take account of the following anti-pattern [24] for Insertion mutation operation.
  - *Anti-append Early Exit*: This pattern disallows the insertion of return statements at any location except for after the last statement.

  The utility of anti-pattern is not directly used with the semantic-based counterparts. Rather it is used to filter out the trivial solutions.

After calculating the syntactic and semantic features, they are then combined to calculate the final score. We keep Genealogical Similarity and Normalized LCS fixed and used Cosine Similarity, Normalized Edit Distance, Jaccard Similarity Index one at a time. So, we evaluated total 3 combinations of features. The first one was **Com-CS**, the combination of Genealogical Similarity, Normalized LCS and Cosine Similarity. The second one was **Com-NED**, the combination of Genealogical Similarity, Normalized LCS and Normalized Edit Distance. And the final one was **Com-JS**, the combination of Genealogical Similarity, Normalized LCS and Jaccard Similarity Index.

We calculate the score of the patches in the following way:

$$patch\_score = gen\_s(f_n, f_e) + n\_lcs(f_n, f_e) + simi(f_n, f_e) \quad (6)$$

Here, $simi(f_e, f_n)$ stands for Cosine Similarity, Normalized Edit Distance, Jaccard Similarity Index, and Jaccard Similarity Index which we plugged into the equation one at a time.

D. Patch Validation

Patch validation validates the patch that is the correct repair of the faulty node. This step executes negative and positive test cases for validating the patches.

IV. Experiment

We implemented the technique addressed in this paper in Java. It uses the Eclipse JDT parser for manipulating AST. It uses the GZoltar [25] tool to localize the faulty codes. GZoltar tool is used with Ochiai algorithm [26] to calculate the suspicious value of the *Expression* and *Expression-Statement* for being a faulty node. The pseudocode of our technique is given in 1.

Algorithm 1: High-level pseudocode for our technique.

Input : Program *P* to be repaired; Set of positive test cases, *PosT*; Set of negative test cases, *NegT*; Set of augmented mutation operation, *AugOp*.
Output: Repaired program variant

```
RepairedProgramVariants ← ∅
φ ← localize_fault(P, PosT, NegT)
forall faultyNode ∈ φ do
    θ ← getFixingElement()
    N ← getParent(faultyNode)
    forall fixingIngredient ∈ θ do
        OP ← getMutationOperation(N, fixingIngredient)
    end
end
forall fixingIngredient ∈ θ & OP ∈ AugOp do
    syntacticSimi ← calculateSyntacticSimi(faultyNode, fixingIngredient, OP)
    semanticSimi ← calculateSemanticSimi(faultyNode, fixingIngredient, OP)
    finalScore ← combine(syntacticSimi, semanticSimi)
    candidatePatches ← {faultyNode, fixingIngredient, finalScore}
end
sortedCandidatePatches ← sort(candidatePatches)
forall patch ∈ sortedCandidatePatches do
    programVariant ← insert(faultyNode, fixingIngredient, AugOp)
    RepairedProgramVariants ← validate(programVariant, PosT, NegT)
end
return RepairedProgramVariants
```

A. Research Question

As we discussed in the earlier section, the existing patch prioritization technique can be categorized into 3 broad categories. Recent work [5] introduced the technique for combining syntactic and semantic similarity for patch prioritization, worked at the *Expression* level bug and used the *Replacement* mutation operation to generate patches. In this study, we evaluate the technique of combining syntactic and semantic similarity to observe its impact on patch prioritization using the *Insertion* mutation operation. Our evaluation aims to answer the following research questions:

- **RQ1:** What is the impact of combining syntactic and semantic similarity while using *Insertion* mutation operation?
- **RQ2:** How different syntactic and semantic similarity features impact patch prioritization?

B. Evaluation

We evaluated our technique in IntroClassJava Benchmark [23]. The benchmark contains 297 bugs from 6 projects. For evaluation, the following are examined:

- **Median Rank of the Correct Patch:** The lower the median rank, the better the approach is [10].
- **Precision:** If a technique can rank the correct solution before the incorrect plausible one, it achieves higher precision. [22]
- **Rank of the first correct solution:** The approach that ranks the first correct solution higher, considered as more efficient. [10].

C. Experiment Settings

To understand the impact of similarities, the approach need to compared with patch prioritization techniques that use semantic and syntactic similarity individually. Therefore, this study further implements semantic or syntantic similarity based patch prioritization approaches using metrics discussed in Section III-C. The techniques are described below:

- **Semantic Similarity based Approach (SSBA):** It uses only semantic similarity metrics namely genealogical to prioritize patches. Also the anti-pattern is also used to filter out the incorrect patches.
- **LCS based Approach (LBA):** It is a syntactic similarity based approach that prioritizes patches using only normalized LCS score.
- **Cosine Similarity based Approach (CSBA):** It is another syntactic similarity based approach that prioritizes patches using only normalized Cosine score.
- **Jaccard Similarity Index based Approach (JSBA):** It is also a syntactic similarity based approach that uses only Jaccard Similarity Index score
- **Normalized Edit Distance based Approach (NBA):** It is also another syntactic similarity based approach that prioritizes patches using only normalized Edit Distance score.

All of these five patch prioritization approaches follow the same repairing process as the combined ones. It ensures that the observed effects occurred due to varied similarities used in patch prioritization.

D. Result Analysis

The approach repairs 6 bugs from IntroClassJava benchmark [23]. For the repaired bugs, no plausible patch is generated before the first correct solution. Thus, the approach achieves a precision of 100%. The implementation is publicly available here.

In our study, we were able to repair 6 bugs from the IntroClassJava benchmark. The results are shown in the table I

Figure 2 shows the first correct patch rank of different approach. The lower the rank is, the better the approach. Figure 3 shows the rank distribution of Com-CS, Com-NED, Com-JS, SSBA, LBA, CSBA, JSBA, and NBA. Log transformation

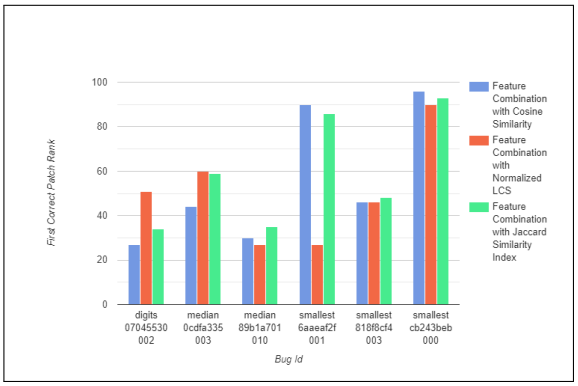


Fig. 2: Impact of different combination of features on patch rank

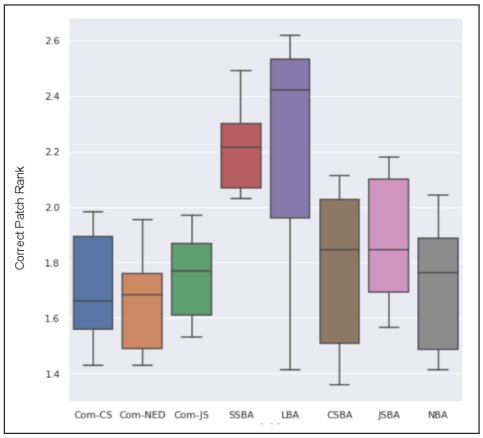


Fig. 3: Comparison of Correct Patch Rank among various Approach

is used in this figure as the data range is high. In terms of median rank, Com-CS, Com-NED, and Com-JS outperform SSBA, LBA, CSBA, JSBA and NBA. The ranks are 45, 48.5, 53.5, 163.5, 177, 55, 55, 58.5 respectively. The reason is combination of features incorporate information from multiple domains (both textual similarity and code meaning). The correctness of the generated patches are checked using the publicly available results of CapGen [22] except for the bug *smallest_cb243beb_000*. CapGen doesn't contain the fix for that bug. The study [6] that shows that combining both semantic and syntactic similarities works better than the syntactic or semantic similarities alone, was able to fix 22 bugs using Replacement Mutation Operation. In this paper we evaluate the impact if we use Insertion Mutation Operation. So, in total 25 bugs have been solved from the *IntroClassJava* benchmark [23] using the approach of combining syntactic and semantic similarities.

V. Conclusion

In this paper, a patch prioritization technique is evaluated for combining syntactic and semantic similarity for automated program repair. Our approach uses different combinations of syntactic and semantic matrices to see their impact on patch

Bug Id	With Cosine Similarity		With Normalized Edit Distance		With Jaccard Similarity Index	
	Total Patches	Correct Patch Rank	Total Patches	Correct Patch Rank	Total Patches	Correct Patch Rank
digits_07045530_002.java	57	27	78	51	57	34
median_0cdfa335_003.java	137	44,68	170	60	137	59, 63
median_89b1a701_010.java	145	30	185	27	145	35
smallest_6aaeaf2f_001.java	197	90	226	27	197	86
smallest_818f8cf4_003.java	199	46	178	46	199	48
smallest_cb243beb_000.java	168	96	189	90	168	93

TABLE I: Results of the Approach of Combining Features on IntroClassJava Benchmark

Bug Id	SSBA		LBA		CSBA		JSBA		NBA	
	TP	CPR	TP	CPR	TP	CPR	TP	CPR	TP	CPR
digits_07045530_002.java	326	107	357	342	58	23	58	42	82	67
median_0cdfa335_003.java	652	110, 120	682	287, 340	138	42, 92	138	52, 85	175	81
median_89b1a701_010.java	660	299	645	26	146	26	146	37	189	26
smallest_6aaeaf2f_001.java	817	170	870	113	198	130	198	140, 151	233	26
smallest_818f8cf4_003.java	779	157, 175	851	48, 414	200	70	200	58	181	50
smallest_cb243beb_000.java	589	310	631	241	168	124	168	121	190	110

TABLE II: Results of the Approach using Only Semantic or Syntactic Features Alone on IntroClassJava Benchmark (TP = Total Patches, CPR = Correct Patch Ranks)

prioritization while using the Insertion mutation operation for patch generation. It uses *Genealogical Similarity*, *Normalized LCS*, *Cosine Similarity*, *Jaccard Similarity Index*, *Normalized Edit Distance* to calculate the context similarity score. It also considers *Anti-Patterns* to filter out the incorrect plausible patches. The approach observes that *Cosine Similarity* and *Normalized Edit Distance* along with *Genealogical Similarity* and *Normalized LCS* rank the correct patch higher. This approach has been able to solve 6 bugs from IntroClassJava benchmark [23]. It achieves a precision of 100% for solving those bugs.

REFERENCES

[1] Tom Britton, Lisa Jeng, Graham Carver, Paul Cheak, and Tomer Katzenellenbogen. 2013. Reversible debugging software. Judge Bus. School, Univ. Cambridge, Cambridge, UK, Tech. Rep (2013).

[2] M. Monperrus, "Automatic software repair: a bibliography," ACM Computing Surveys (CSUR), vol. 51, no. 1, p. 17, 2018.

[3] E. T. Barr, Y. Brun, P. Devanbu, M. Harman, and F. Sarro, "The plastic surgery hypothesis," in ESEC/FSE, 2014, pp. 306–317.

[4] S. Sumi, Y. Higo, K. Hotta, and S. Kusumoto, "Toward improving graftability on automated program repair," in ICSME, 2015, pp. 511–515.

[5] M. Asad, K. K. Ganguly and K. Sakib, "Impact Analysis of Syntactic and Semantic Similarities on Patch Prioritization in Automated Program Repair," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 328-332, doi: 10.1109/ICSME.2019.00050.

[6] Z. Chen and M. Monperrus, "The remarkable role of similarity in redundancy-based program repair," Computing Research Repository (CoRR), vol. abs/1811.05703, 2018.

[7] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "Genprog: A generic method for automatic software repair," IEEE Transactions on Software Engineering (TSE), vol. 38, no. 1, pp. 54 –72, 2012

[8] Y. Pei, Y. Wei, C. Furia, M. Nordio, and B. Meyer, "Code-based automated program fixing," in International Conference on Automated Software Engineering (ASE), 2011, pp. 392 –395.

[9] Dongsun Kim, Jaechang Nam, Jaewoo Song, and Sunghun Kim. 2013. Automatic patch generation learned from human-written patches. In ICSE'2013. IEEE Press, 802–811.

[10] Xuan Bach D Le, David Lo, and Claire Le Goues. 2016. History Driven Program Repair. In SANER'2016, Vol. 1. IEEE, 213–224.

[11] Hoang Duong Thien Nguyen, Dawei Qi, Abhik Roychoudhury, and Satish Chandra. 2013. SemFix: program repair via semantic analysis. In ICSE'2013. IEEE Press, 772–781.

[12] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically finding patches using genetic programming. In ICSE'2009. IEEE Computer Society, 364–374.

[13] V. Debroy and W. Wong, "Using mutation to automatically suggest fixes for faulty programs," in International Conference on Software Testing, Verification and Validation (ICST), 2010, pp. 65 –74.

[14] D Le Xuan-Bach, Chu Duc-Hiep, Lo David, Goues Claire, Le, and Willem Visser. 2017. S3: Syntax- and Semantic-Guided Repair Synthesis via Programming by Examples. In FSE'2017. ACM, to appear.

[15] Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, and Westley Weimer. 2012. A systematic study of automated program repair: Fixing 55 out of 105 bugs for 8 each. In ICSE'2012. IEEE, 3–13.

[16] Westley Weimer, Zachary P Fry, and Stephanie Forrest. 2013. Leveraging program equivalence for adaptive program repair: Models and first results. In ASE'2013. IEEE, 356–366.

[17] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "Genprog: A generic method for automatic software repair," IEEE Transactions on Software Engineering (TSE), vol. 38, no. 1, pp. 54 –72, 2012

[18] R. K. Saha, Y. Lyu, H. Yoshida, and M. R. Prasad, "Elixir: Effective object-oriented program repair," in Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 648–659, IEEE, 2017.

[19] J. Jiang, Y. Xiong, H. Zhang, Q. Gao, and X. Chen, "Shaping program repair space with existing patches and similar code," in Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp. 298–309, ACM, 2018.

[20] Q. Xin and S. P. Reiss, "Leveraging syntax-related code for automated program repair," in Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 660–670, IEEE, 2017.

[21] R. Just, D. Jalali, and M. D. Ernst, "Defects4J: A database of existing faults to enable controlled testing studies for Java programs," in ISSTA, 2014, pp. 437–440.

[22] M. Wen, J. Chen, R. Wu, D. Hao, and S.-C. Cheung, "Context-aware patch generation for better automated program repair," in Proceedings of the 40th International Conference on Software Engineering (ICSE), pp. 1–11, ACM, 2018.

[23] T. Durieux and M. Monperrus, IntroClassJava: A Benchmark of 297 Small and Buggy Java Programs. PhD thesis, Universite Lille 1, 2016.

[24] Shin Hwei Tan, Hiroaki Yoshida, Mukul R Prasad, and Abhik Roychoudhury. 2016. Anti-patterns in search-based program repair. In International Symposium on Foundations of Software Engineering. ACM, 727–738.

[25] J. Campos, A. Riboira, A. Perez, and R. Abreu, "GZoltar: an eclipse plug-in for testing and debugging," in ASE, 2012, pp. 378–381.

[26] R. Abreu, P. Zoeteweij, and A. J. C. v. Gemund, "An evaluation of similarity coefficients for software fault localization," in Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing, ser. PRDC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 39–46.

Two-Stage AST Encoding for Software Defect Prediction

Yanwu Zhou[a], Lu Lu[a,*], Quanyi Zou[b], Cuixu Li[c]
[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[b] School of Software Engineering, South China University of Technology, Guangzhou, China
[c] Guangdong Meiweixian Flavoring Foods Co.,Ltd., Zhongshan, China
*Corresponding author email: lul@scut.edu.cn

***Abstract*—Software defect prediction (SDP) can find potential containing defect modules, which assists software developers in allocating limited test resources more efficiently. Because traditional software features fail to capture the semantics of source code, various studies have turned to extracting deep learning features. Existing related approaches often parse the program source code into Abstract Syntax Trees (ASTs) for further processing. However, most of these approaches ignore AST nodes' hierarchical and position-sensitive structure. To overcome the aforementioned issues, a two-stage AST encoding (TSE) method is proposed in this paper for software defect prediction. Experiments on eight Java open-source projects showed that our proposed SDP method outperforms several traditional methods and state-of-the-art deep learning methods in terms of F-measure and MCC.**

***Index Terms*—software defect prediction, abstract syntax tree (AST), two-stage encoding, positional encoding**

I. Introduction

As software evolves, the scale and complexity of the system grow dramatically. Under limited time and resource constraints, software becomes more prone to defect [1]. Software defect prediction (SDP) is a promising technology for improving software reliability by detecting program defect modules and prioritizing testing efforts [2]. The goal of SDP is to train a defect predictor that classify code instances as defective or not. It is a requirement for SDP to create defect-distinguishable software representations [3]. Software metrics have been proposed and broadly used in SDP [4]. However, Software metrics are hand-crafted by software specialists and most software metrics focus on the statistical aspects of code, ignoring the semantic characteristics of code.

In recent years, researchers have begun to leverage deep neural networks to exact software features from source code [5]. Related works have revealed that Abstract Syntax Trees (ASTs) are suitably representative of programs' well-defined syntax [6]–[8]. It's common practice to parse source code files into ASTs and then convert into token vectors by traversing the ASTs.

Pre-order traversal is adopted by most studies for AST conversion, treating all nodes as the same level and creating a token corpus for word embedding technology. It has two drawbacks: 1. Not all the nodes are at the same level. The coarse-grained information of node needs to be supplemented by the fine-grained information of its child nodes. 2. If only preorder traversal is employed for AST, the tree structure's positional information will be lost. Furthermore, there is a sequential positional relationship between nodes at the same depth.

In order to tackle the above-mentioned first drawback, it is feasible to decompose the AST into non-overlapping subtrees according to node granularity. For example, Zhang et al. propose ASTNN, which divides the AST into subtrees at the statement level and handles the subtree interior and subtree sequence independently [9]. The second drawback can be addressed by including the additional position information of the tree structure.

In this paper, we mark two types of nodes with different granularities, named ordinary nodes and block nodes, and then execute a two-stage encoding, according to the decomposition strategy. In the first stage, the word embedding and positional embedding of ordinary nodes under the subtree rooted by the block node are aggregated to the block node through the self-attention mechanism to represent the encoding of the block node. In the second stage, the encodings of the block nodes in an AST are collected. The tree structure of block nodes is retained, and the encodings are fed into a Tree-based LSTM network to generate the final AST representation for software defect prediction.

In summary, the main contributions of this paper are listed as follows:

1) We propose an AST decomposition strategy that marks AST nodes as two types of nodes according to the hierarchy of the AST.
2) We apply the strategy on a two-stage bottom-up AST encoding for software defect prediction. Experimental results indicate that our method outperforms other software defect prediction models in terms of F-measure and MCC.

II. Related work

A. Code Representation in Software Defect Prediction

The representation of software code is a critical part of software defect prediction. Wang et al. proposed to parse the source code into ASTs, and then encode it into numerical vectors as code representation [6]. Li et al. concatenate deep

DOI reference number: 10.18293/SEKE2022-039

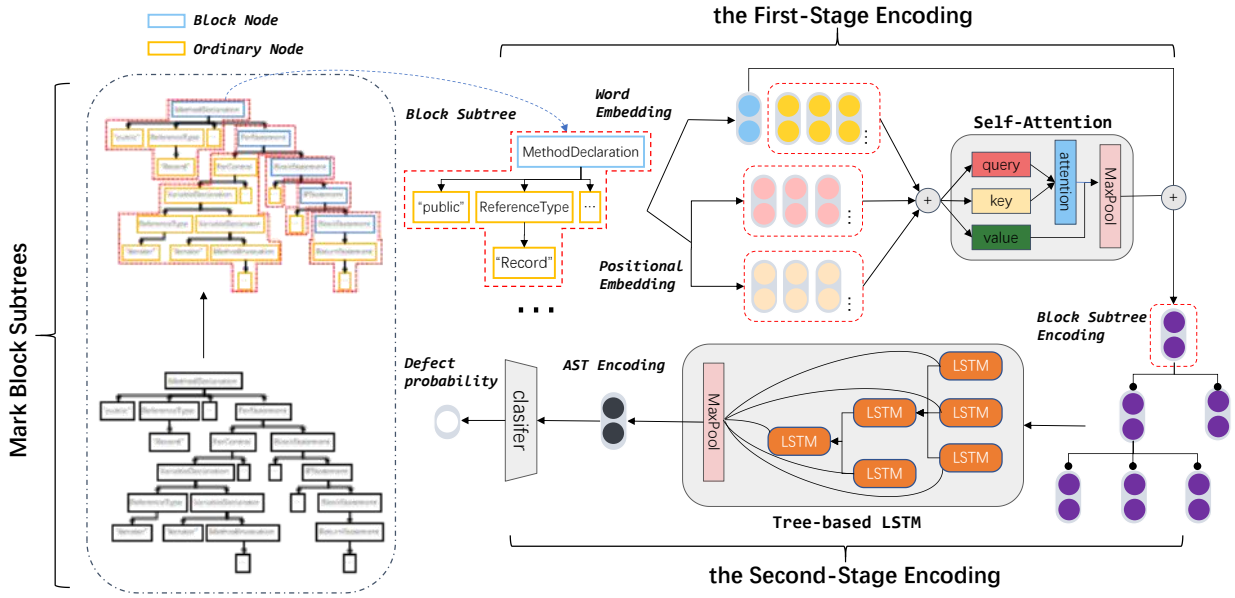


Fig. 1. The overall framework of TSE.

semantic features with code metrics to augment the information of code [7]. In addition to the code representation based on AST, Phan et al. construct the Control Flow Graph (CFG) of the program at the level of assembly instruction [10]. The effectiveness of code visualization is demonstrated by Chen et al., who convert each code file in the program into a two-dimensional image. [11].

B. Deep Learning in Software Defect Prediction

There has been a slew of new deep learning applications for SDP. Wang et al. built the Deep Believe Network (DBN) for extracting features [6]. Li et al. considered that CNN can seize local patterns of sources code more effectively [7]. Xu et al. employed graph neural networks to capture the latent defect information [12]. Wang et al. proposed a modular tree network to dig the semantic differences among different types of AST substructures which shows the advantage of more elaborate structure information extraction [13].

III. Method

In this section, we elaborate on our method in detail. Figure 1 demonstrates the overall framework of our TSE method. Before the encoding stages, the code file is parsed into an AST, and then marked and converted into vectors. In the first stage, the self-attention layer is utilized to aggregate the word embeddings and positional encodings of all ordinary nodes to construct the block node encodings. In the second encoding stage, the block node encodings are fed into a Tree-based LSTM and produce the AST's final encoding by max pooling. Finally, the probability that the code file is defective is output via a fully-connected layer.

A. Marking AST nodes and Converting into Vectors

The open-source code tool, *javalang*[1], is used to parse the code files into ASTs under the token granularity. And then the block nodes and ordinary nodes are marked during the pre-order traversal of the AST. The set of block nodes and ordinary nodes are difined as B and O, separately. Block nodes are listed in the Table I and ordinary nodes are chosen according to Wang's work [6]. Note that the attribution strings of the AST node are also regarded as ordinary nodes. Most of the block nodes chosen create a local scope with specific context. The nodes in the *IfStatement* block, for example, are in the specific context of conditional judgment; hence, the meaning of the nodes of *IfStatement* may differ while in other statement blocks. There is a particular node in the block nodes called *StatementExpression*, which is a statement-level node. This node can further disassemble the statement block and tackle the problem of an excessively large subtree.

An AST T is defined as a collection of its block subtrees:

$$T = [bT_1, bT_2, ..., bT_n] \tag{1}$$

Each block subtree is defined as:

$$bT_i = [b_i, [oT_{i1}, oT_{i2}...]] \tag{2}$$

where $b_i \in B$ is the root node of the block subtree, oT_{ij} is the ordinary subtree under bT_i. An ordinary subtree is denoted as:

$$oT_{ij} = [o_{ij1}, o_{ij2}, ...] \tag{3}$$

where $o_{ijk} \in O$.

After acquiring the marked AST, we utilize word2vec [14] technique to convert block nodes and normal nodes into E-dimensional vectors, denoted as x_i^b and x_{ijk}^o, respectively.

[1]https://github.com/c2nes/javalang

B. First Stage: Aggregating Ordinary Node Encodings to The Block Node Encoding

In the first stage, the self-attention mechanism is employed to aggregate their information onto the block nodes as the block node encodings:

$$Atten(X) = Atten(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

$$Q = W^Q x_q, K = W^K x_k, V = W^V x_v \quad (5)$$

where W^Q, W^K and W^V are weight matrices for queries, keys and values, and d_k is the dimension of x_k. In the case of self-attention, x_q, x_k and x_v are identical. The self-attention mechanism can flexibly assign the attention weights of the same nodes in different contexts and mask the difference in the number of nodes in a subtree. Self-attention mechanism is not sensitive to the position of nodes. However, the positional relationship of nodes in the subtree could be a significant factor in defect prediction. Therefore, we additionally traverse the block subtree hierarchically and record the positional information of each ordinary node which consists of the depth of the node in the block subtree and the sequence number of the node at the depth. Through an embedding layer, their corresponding positional vectors can be represented separately as x^{depth} and x^{num}.

Given bT_i with N_i nodes, We can obtain the embedding sequence:

$$X_i^{bT} = [x_i^b, x_{i11}^o + x_{i11}^{depth} + x_{i11}^{num}, ..., x_{ijk}^o + x_{ijk}^{depth} + x_{ijk}^{num}] \quad (6)$$

Then the encoding of a subtree $en_i^{bT} \in R^D$ is calculated by:

$$en_i^{bT} = (W^{en})^T x_i^b + Maxpool(Atten(X_i^{bT}|x_i^b)) + b^{en} \quad (7)$$

where D is the encoding dimension, $W^{en} \in R^{E \times D}$ is the weight matrix, b^{en} is the bias term, $Maxpool(.)$ is the max pool layer with the kernel size N_i, $Atten(.)$ is the self-attention layer, $X_i^{bT}|x_i^b$ is the sequence X_i^{bT} excluding x_i^b.

C. Second Stage: Aggregating Block Node Encodings to The AST Encoding

In the first encoding stage, the encodings of all block nodes in an AST are collected:

$$X^T = [en_1^{bT}, ..., en_n^{bT}] \quad (8)$$

Tree-based LSTM is adopted as the encoder to acquire the final AST's encoding, preserving more information about the AST's tree structure. Since all ordinary nodes in an AST have been processed in the first encoding stage, the scale of the AST has been considerably reduced, hence avoiding gradient vanishing induced by long-term dependencies. Specifically, we utilize ChildSum Tree-LSTM [15] to calculate the AST's final encoding by max-pooling the hidden states of block nodes:

$$en^T = Maxpool(TreeLSTM(X^T)) \quad (9)$$

Then we simply use a fully-connected layer as the defect classifier. Cross entropy loss is adopted for optimization.

TABLE I
THE CHOSEN TYPES OF BLOCK NODES

Block Nodes	ClassDeclaration, InnerClassDeclaration, MethodDeclaration, ConstructorDeclaration, BlockStatement, ForStatement, WhileStatement, SwitchStatement, IfStatement, DoStatement, **StatementExpression**

IV. Experimental Settings

A. Evaluated Projects and Datasets

To evaluate the the effectiveness of our TSE approach, we choose publicly available projects from PROMISE repository [16]. Specifically, eight open-source Java projects are exploit in our experiments, which are *ant*, *camel*, *jedit*, *log4j*, *lucene*, *poi*, *xalan* and *synapse*.

B. Evaluation

In this paper, the popular evaluation indicator in SDP, F-measure and MCC, are adopted to evaluate our proposed method. Specifically, F-measure is a harmonic mean of Precision and Recall and MCC is a relatively balanced measure considering diverse indicators, which are calculated by the following equations:

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \quad (10)$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (11)$$

$$MCC = \frac{TP * TN - FP * FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \quad (12)$$

where TP, FN, FP and TN can be derived by the confusion matrix.

C. Baseline Setting

In this paper, our TSE method are compared with the following methods:

1) LR: A traditional method using logistic regression classifier on handcraft defect metrics.
2) SVM: A traditional method using SVM classifier with the Gaussian kernel on handcraft defect metrics.
3) DBN: A deep learning method employing a standard DBN model to extract semantic features from AST for SDP [6].
4) LSTM: A deep learning method using LSTM to capture semantic representations for SDP.
5) CNN+: An enhanced CNN-based method by combining traditional feature and deep feature for SDP [7].

D. Parameters Setting

10-fold cross validation is used to split dataset and each training task is repeated 10 times in the experiment. The AST tokens and the positional vectors are embedded into 100-dimensional vectors; The query, key, and value dimensions are all set to 100 in the self-attention layer; The hidden dimension of the ChildSum Tree-LSTM is 100.

TABLE II
F-MEASURE AND MCC VALUE FOR TSE VERSUS BASELINE METHODS

Task	LR		SVM		DBN		LSTM		CNN+		TSE	
	F-measure	MCC	F-measure	MCC	F-measure	MCC	F-measure	MCC	F-measure	MCC	F-measure	MCC
ant	**0.568**	0.427	0.561	0.417	0.371	0.220	0.533	0.214	0.566	**0.437**	0.505	0.403
camel	0.352	0.167	0.356	0.177	0.359	0.171	0.370	0.199	0.335	0.180	**0.501**	**0.389**
jedit	0.552	0.388	0.534	0.365	0.504	0.174	0.551	0.345	0.561	**0.451**	**0.572**	0.438
log4j	0.409	**0.301**	0.338	0.250	0.611	0.132	**0.633**	0.162	0.593	0.219	0.628	0.255
lucene	0.530	0.161	0.600	0.141	0.574	0.161	0.605	0.171	0.643	0.236	**0.654**	**0.243**
poi	0.589	0.238	0.791	0.479	0.626	0.310	0.752	0.314	0.748	0.328	**0.778**	**0.361**
xalan	0.532	0.101	0.545	0.113	0.592	0.077	0.602	0.124	0.635	0.176	**0.653**	**0.184**
synapse	0.506	**0.294**	0.480	0.292	0.413	0.163	0.344	0.118	0.369	0.257	**0.550**	0.282
Average	0.501	0.249	0.523	0.270	0.506	0.173	0.549	0.205	0.556	0.275	**0.612**	**0.314**

V. EXPERIMENTAL RESULTS

Table II record comparison results of the F-measure and MCC indicators for the TSE method versus other baseline methods. According to the last second line of the two tables, our proposed TSE achieves F-measure as 0.612 and MCC as 0.314 on average value and obtains the best average value on the two indicators. Compared to the traditional machine learning methods, i.e., LR and SVM, which utilize statistical defect metrics, our TSE method has an average improvement of 22.2% and 16.9% in the F-measure indicator, and 26.1% and 16.0% in the MCC indicator. Compared to the deep learning methods, i.e., DBN, LSTM and CNN+, our TSE method has an average improvement of 18.8%, 11.3% and 10.1% in terms of F-measure, and 81.4%, 52.9% and 14.0% in terms of MCC. The above experiments suggest that TSE outperforms traditional machine learning methods and AST-based semantic extracting methods on eight separate projects.

VI. CONCLUSION

In this paper, a two-stage AST encoding method is proposed, which employs a bottom-up encoding to learn the semantic information software defect prediction. The main advantages of TSE are 1. executing the encoding following the natural hierarchy of ASTs. 2. combining the tree positional encoding to augment the structural information of ASTs. The performance of TSE is evaluated by comparing it with traditional methods and state-of-the-art deep learning methods in terms of F-measure and MCC. Experimental results show that TSE achieves better performance versus all baseline methods. In future work, we will verify the performance of our method on other programming languages and repositories.

ACKNOWLEDGMENT

This work was supported in part by the Zhongshan Produce and Research Fund, PR China under grant no. 210602103890051.

REFERENCES

[1] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?," *Software Quality Journal*, vol. 26, no. 2, pp. 525–552, 2018.

[2] H. K. Dam, T. Tran, T. Pham, S. W. Ng, J. Grundy, and A. Ghose, "Automatic feature learning for predicting vulnerable software components," *IEEE Transactions on Software Engineering*, vol. 47, no. 01, pp. 67–85, 2021.

[3] H. Wang, W. Zhuang, and X. Zhang, "Software defect prediction based on gated hierarchical lstms," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 711–727, 2021.

[4] M. Halstead, "Elements of software science (operating and programming systems series)," 1977.

[5] N. Zhang, S. Ying, K. Zhu, and D. Zhu, "Software defect prediction based on stacked sparse denoising autoencoders and enhanced extreme learning machine," *IET Software*, 2021.

[6] S. Wang, T. Liu, J. Nam, and L. Tan, "Deep semantic feature learning for software defect prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1267–1293, 2018.

[7] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 318–328, IEEE, 2017.

[8] X. Zhou and L. Lu, "Defect prediction via lstm based on sequence and tree structure," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 366–373, IEEE, 2020.

[9] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 783–794, IEEE, 2019.

[10] A. V. Phan, M. Le Nguyen, and L. T. Bui, "Convolutional neural networks over control flow graphs for software defect prediction," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 45–52, IEEE, 2017.

[11] J. Chen, K. Hu, Y. Yu, Z. Chen, Q. Xuan, Y. Liu, and V. Filkov, "Software visualization and deep transfer learning for effective software defect prediction," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 578–589, 2020.

[12] J. Xu, F. Wang, and J. Ai, "Defect prediction with semantics and context features of codes based on graph representation learning," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 613–625, 2020.

[13] W. Wang, G. Li, S. Shen, X. Xia, and Z. Jin, "Modular tree network for source code representation learning," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–23, 2020.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[15] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[16] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: An empirical study on defect prediction," in *2013 ACM/IEEE international symposium on empirical software engineering and measurement*, pp. 45–54, IEEE, 2013.

An efficient discrimination discovery method for fairness testing

Shinya Sano
Dept. of Info. and Comp. Sci.
Keio University
Yokohama, Japan
sanoshin@doi.ics.keio.ac.jp

Takashi Kitamura
Nat. Inst. of Advanced Industrial Science and Technology
Tokyo, Japan
t.kitamura@aist.go.jp

Shingo Takada
Dept. of Info. and Comp. Sci.
Keio University
Yokohama, Japan
michigan@ics.keio.ac.jp

***Abstract*—With the increasing use of machine learning software in our daily life, software fairness has become a growing concern. In this paper, we propose an individual fairness testing technique called KOSEI. Individual fairness is one of the central concepts in software fairness. Testing individual fairness aims to detect individual discriminations included in the software. KOSEI is based on AEQUITAS by Udeshi et al., a landmark fairness testing technique featuring a two-step search strategy of global and local search. KOSEI improves the local search part of AEQUITAS, based on our insight to overcome the limitations of the local search of AEQUITAS. Our experiments show that KOSEI outperforms AEQUITAS by orders of magnitude. KOSEI, on average, detects 5,084.8% more discriminations than AEQUITAS, in just 7.5% of the execution time.**

***Index Terms*—software testing, algorithm fairness, machine learning**

I. INTRODUCTION

With the increasing use of machine learning (ML) software in our daily life, software fairness has become a growing concern. A famous example is the COMPAS software, which computes risk-assessment scores for recidivism of defendants. It assists in the sentencing process; however, the software makes biased and discriminatory mistakes [1]. For example, the software is likely to falsely rate more black defendants to be risky than white defendants.

Individual fairness is a key concept in software fairness. It is often referred to with the concept of individual discrimination. Individual discrimination occurs when ML software gives different results to two similar individuals that differ in protected attributes. Protected attributes represent attributes often tied to social bias, e. g., gender, race, or age.

Testing ML software for individual fairness is one approach to validate software fairness. It has recently been under active investigation, with various algorithms being proposed. For example, some use random sampling techniques [1], some use probabilistic searches [2], some apply the technique of gradient-based adversarial sampling [3], some apply symbolic execution [4], and some apply constraint solving [5].

In this paper, we tackle black-box individual fairness testing, proposing a technique called KOSEI[1] (Keeper Of Systematic Equality Investigation). KOSEI is based on AEQUITAS, a landmark technique for black-box individual fairness testing, developed by Udeshi et al. [2]. A characteristic feature of AEQUITAS is that its testing algorithm is structured into two phases of global and local search to leverage the robustness of ML algorithms for efficiently detecting discriminations. The two-phased structure of the algorithm has become the basis of many crucial algorithms (e. g., [3], [4]). Focusing on improving the second phase of the algorithm (called local search), KOSEI aims to obtain a higher detecting ability of individual discriminations. Our evaluation of KOSEI shows that KOSEI outperforms AEQUITAS by orders of magnitude. KOSEI detects, on average, 5,084.8% more discriminations than AEQUITAS in just 7.5% of the execution time. Our evaluation also confirms that our technical refinements realize the performance gain.

The paper is organized as follows. Section II reviews individual fairness testing and AEQUITAS by Udeshi et al. [2]. In Section III, we propose our method, KOSEI. In Section IV, we evaluate KOSEI through experiments. We discuss threats to validity in Section V. Section VI discusses related work. Finally, Section VII concludes this paper.

II. BACKGROUND

This section reviews individual fairness testing (referring to [6]) and AEQUITAS [2].

A. Individual fairness testing

Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of *attributes* (or *parameters*), for $n \in \mathbb{N}$. We use p_i to indicate the i-th attribute in P. Each attribute $p_i (\in P)$ is associated with a set of *values*, called the *domain* of p_i, denoted by $Dom(p_i)$, such that $(Dom(p_i))_{i \in n}$ is pairwise disjoint. The input space $\mathbb{I}$ of a set of attributes P is the Cartesian product of the domains of $p_1, p_2 \cdots p_n (\in P)$, i. e., $\mathbb{I} = Dom(p_1) \times Dom(p_2) \times \cdots \times Dom(p_n)$. An element I of $\mathbb{I}$ is a *data item*, which we may also call a *test case*. We use I_k as the value of the k-th attribute of input $I \in \mathbb{I}$. We also introduce $P_{protected} \subseteq P$ as the set of protected attributes (e. g., gender, race, age). We interpret a ML classifier, whose input space is $\mathbb{I}$, as function f; i. e., we use $f(I)$ to denote the result (i. e., decision) that the trained classifier f makes for input I.

DOI reference number: 10.18293/SEKE2022-064

[1]Japanese word which means fairness

Definition 1 (Discriminatory data and Fairness [2]): Let f be a classifier under test, γ be the pre-determined discrimination threshold (e.g. chosen by the user), and $I \in \mathbb{I}$. Assume $I' \in \mathbb{I}$ such that there exists a non-empty set $Q \subseteq P_{protected}$ and for all $q \in Q$, $I_q \neq I'_q$ and for all $p \in P \setminus Q$, $I_p = I'_p$. If $|f(I) - f(I')| > \gamma$, then I is called a discriminatory data item of the classifier f and is an instance that manifests the violation of (individual) fairness in f.

Example 1: We use the Census Income (aka, Adult) dataset [7] as the running example. Its task is to predict if the income of an adult exceeds $\$50,000$ per year. The dataset contains 32,561 training instances with 13 attributes each. The following example shows a numeric-represented data instance x:

$$x : [4, 0, 6, 6, 0, 1, 2, 1, \dot{1}, 0, 0, 40, 30]$$

The first attribute represents 'age', whose domain is {0..9} (where value '4', for instance, means age from 40 to 49 years); the ninth represents 'gender', whose domain is 'male (0)' and 'female (1)'. The running example considers 'gender' (with a dot) as the protected attribute.

Classifier f inputs a data item of the Census data set and returns '1' if the income of an adult exceeds $\$50,000$ per year, and '0' otherwise. According to Definition 1, the data item x is discriminatory, for the classifier f, the protected (i. e., 'gender') attribute, and $\gamma = 0$, if $f(x) \neq f(x')$ with

$$x' : [4, 0, 6, 6, 0, 1, 2, 1, \dot{0}, 0, 0, 40, 30].$$

Observe that x' differs from x only in the 'gender' attribute.

Algorithm 1: Local Search Of Aequitas: $local\_search(D_{global}, f, limit)$

Data: discriminatory data (D_{global}), classifier(f), local iteration limit ($limit$)
Result: Discriminatory data (D_{local})

1 **Procedure** *local_search($D_{global}, f, limit$)*
2 $\quad \sigma_{pr}[p] \leftarrow \frac{1}{|P'|}$ for all $p \in P_{non\_protected}$
3 $\quad \sigma_v[p] \leftarrow 0.5$ for all $p \in P_{non\_protected}$
4 $\quad$ **for** $I \in D_{global}$ **do**
5 $\quad\quad$ **for** i *in* $(0, limit)$ **do**
6 $\quad\quad\quad$ // apply perturbation to I
7 $\quad\quad\quad$ Select $p \in P_{non\_protected}$ with $\sigma_{pr}[p]$
8 $\quad\quad\quad$ Select δ with $\sigma_v[p]$
9 $\quad\quad\quad$ $I[p] \leftarrow I[p] + \delta$
10 $\quad\quad\quad$ **if** *eval_disc(I)* **then** $D_{local} \leftarrow D_{local} \cup \{I\}$
11 $\quad\quad\quad$ update_prob$(I, p, D_{local}, \delta)$
12 $\quad$ **return** $D_{global} \cup D_{local}$

Algorithm 2: $eval\_disc(I)$

Data: A data item (I)
Result: Boolean

1 **Procedure** *eval_disc(I)*
2 $\quad \triangleright$ $\mathbb{I}^{(d)}$ extends I with all values of $P_{protected}$
3 $\quad \mathbb{I}^{(d)} \leftarrow \{I' | \forall p \in P_{non\_protected}. I_p = I'_p\}$
4 $\quad$ **if** $\exists I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma$ **then return** True
5 $\quad$ **else return** False

B. AEQUITAS

We review the algorithm of AEQUITAS, focusing on its two-phase-structured algorithm, perturbation, and an algorithm component called *local search*.

1) Two-phase-structured algorithm: The algorithm of AEQUITAS is structured into the two phases of *global* and *local search*. The algorithm starts with the global search, which randomly searches the input space of a ML classifier. Following the global search, the local search searches nearby the discriminatory data detected in the global search.

This two-phase structure of the algorithm is designed to detect discriminatory data effectively, leveraging the robustness of ML classifiers. The robustness of ML classifiers means that similar prediction is likely to be produced for similar data. So, first, the global search scans the whole input space widely to find different kinds of discriminatory data. Then, the local search searches the vicinities of discriminatory data found in the global search using perturbation to find more discriminatory data.

2) Perturbation : Given a data item, perturbation creates a similar data item by adding small changes to it. For example, Definition 2 gives the perturbation of AEQUITAS.

Definition 2 (perturbation): Perturbation g is a function $g : \mathbb{I} \times (P \setminus P_{protected}) \times \Gamma \rightarrow \mathbb{I}$ where $\Gamma = \{-1, +1\}$. If $I' = g(I, p, \delta)$ where $I \in \mathbb{I}$, $p \in P \setminus P_{protected}$ and $\delta \in \Gamma$, then $I'_p = I_p + \delta$, and $I'_q = I_q$ for all $q \in P \setminus \{p\}$.

Observe that the perturbation adds a change of only -1 or +1 to one attribute of a given data item. As the local search scans the vicinity of discriminatory data passed by the global search, the perturbation stipulates the 'vicinity' of data.

3) Local search : Algorithm 1 shows the local search algorithm of AEQUITAS. The algorithm takes three objects as the input (1) f: the ML classifier under test, (2) D_{global}: a set of discriminatory data (passed from the global search), and (3) $limit$: the number of local iterations. The output is discriminatory data found in the local search.

The algorithm begins with preparing lists σ_{pr} and σ_v. Given an attribute p, $\sigma_{pr}[p]$ shows the probability that p is selected to be perturbed. $\sigma_v[p]$ shows the probability that p is perturbed by $\delta = -1$, while $(1 - \sigma_v[p])$ shows the probability that p is perturbed by $\delta = +1$.

After the initialization of σ_{pr} and σ_v, the algorithm applies a perturbation (line 7 – line 9) to each data item (line 4) in the given discriminatory data (D_{global}) for the $limit$ times (line 5). A perturbation chooses an attribute and the direction of perturbation, respectively based on σ_{pr} (line 7) and σ_v (line 8). For each perturbed data item, the algorithm evaluates if it is discriminatory or not (line 10) using the evaluation function (Algorithm 2). Finally, the data item evaluated as discriminatory is added to D_{local} (line 10). Based on evaluation results, the algorithm updates both σ_{pr} and σ_v

for tuning the probabilities. Three strategies (*random*, *semi-directed*, and *fully-directed*) are provided for this probability update. We do not look into the details in this paper because our proposed technique is not very relevant to the strategies.

Example 2: Consider applying the local search of AEQUITAS to the following data item x:

$$x: \quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1]$$

Suppose the local search algorithm, for the first iteration (i. e., for the first perturbation), chooses the third attribute and the direction of '$+1$' (respectively based on probabilities recorded in σ_{pr} and σ_v). It thus generates the following data item x', and check if it is discriminatory, where note that the value of the third attribute has changed from 26 to 27:

$$x'_1: \quad [7, 4, \underline{27}, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1].$$

For the second iteration, the algorithm works based on this newly generated data item x'_1. Suppose it chooses the 12th attribute and direction of '$+1$'. It thus generates and evaluates the following perturbed data item x'_2:

$$x'_2: \quad [7, 4, \mathbf{27}, 1, 4, 4, 0, 0, \dot{0}, 1, 5, \underline{74}, 1]$$

For the third iteration, where it works on the new data x'_2, suppose it chooses the fifth attribute and direction of '-1'. It thus generates and evaluates the following perturbed data item x'_3:

$$x'_3: \quad [7, 4, \mathbf{27}, 1, \underline{3}, 4, 0, 0, \dot{1}, 1, 5, \mathbf{74}, 1]$$

The algorithm continues similarly until the number of iterations reaches the specified limit.

III. KEEPER OF SYSTEMATIC EQUALITY INVESTIGATION

This section proposes an individual fairness testing technique named Keeper Of Systematic Equality Investigation (KOSEI). KOSEI is based on AEQUITAS; it inherits the two-phase structured search strategy (global and local search) of AEQUITAS but improves the local search algorithm. We state the limitations of AEQUITAS's local search, explain technical details of KOSEI, and discuss its advantages.

A. Limitation of AEQUITAS's local search algorithm

1) AEQUITAS's local search algorithm searches space, where there may be little discriminatory data.: First, the algorithm equally searches vicinities of discriminatory data passed from the global search. Suppose, for example, discriminatory data passed by the global search contain two data d_1 and d_2, in the vicinities of which there are 10 and 1000 discriminatory data, respectively. Even though searching the vicinity of d_2 is likely to find more discriminatory data, the algorithm spends an equal amount of search resources (specified as a limit of local search) on d_1 and d_2.

Second, the algorithm may search not only the vicinities of discriminatory data but also those of non-discriminatory data. Note that the algorithm sequentially applies perturbation to generated data, regardless of whether that data is discriminatory or not. For example, suppose x'_1 (generated by the discriminatory data item x by perturbation) in Example 2 is not discriminatory; the algorithm searches the vicinity of x'_1, by applying perturbation and evaluates the perturbed data item.

2) AEQUITAS's local search may waste search resources by evaluating duplicated data.: First, the perturbation process is based on the *probabilistic choice* of parameter-values; thus, the algorithm may generate a data item that has previously been generated. However, AEQUITAS does not avoid such duplications, which causes its inefficiency.

Specifically, the second is more concerned with the algorithm implementation design. The AEQUITAS implementation of Algorithm 1 (coded in Python) is realized using the basin-hopping optimization function[2]. This implementation design seems to aim to detect discriminatory data efficiently. However, as we observe in our experiments of executing AEQUITAS, the implementation generates quite a few duplicated data during its executions (as is also pointed out in [2]-page 9).

B. Two key mechanisms of our local search

We explain the two key mechanisms in the local search of KOSEI: i. e., perturbation and dynamic update of search space.

1) Perturbation : The concept of perturbation of KOSEI inherits that of AEQUITAS (Definition 2); i. e., the perturbation of KOSEI changes the value of only one attribute of a given data item by $-1/+1$. However, KOSEI uses this perturbation concept differently from AEQUITAS. It applies the perturbation to all the unprotected attributes one by one instead of probabilistically choosing one attribute like in AEQUITAS.

Example 3: The following shows data obtained by applying the perturbation to the data item x:

$$\begin{aligned}
x: &\quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_1: &\quad [\mathbf{6}, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_2: &\quad [\mathbf{8}, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_3: &\quad [7, \mathbf{3}, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
x'_4: &\quad [7, \mathbf{5}, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, 1] \\
&\qquad\qquad \vdots \\
x'_{22}: &\quad [7, 4, 26, 1, 4, 4, 0, 0, \dot{0}, 1, 5, 73, \mathbf{2}]
\end{aligned}$$

The bold numerics indicate attributes to which the perturbation has been applied. For example, x'_1 is generated by perturbating the first (unprotected) attribute (i. e., 'age') of x to the direction of -1; x'_2 is generated by perturbating the first attribute of x to the direction of $+1$; x'_3 is generated by perturbating the second attribute ('work class') of x to the direction of -1. The number of data obtained from one data item x is at most $\#P' * |\delta|$, where $\#P'$ is the number of unprotected attributes and $|\delta| = |\{-1, +1\}| = 2$. Note that some data generated by perturbation will be invalid and hence will be excluded if the perturbed value in the perturbed data is out of its attribute domain.

[2]https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/

Algorithm 3: Local search of KOSEI

```
   Data: Same as Algorithm 1
   Result: Same as Algorithm 1
1  Procedure local_search(D_global, f, limit)
2      // initialize seed data D and D_total
3      // D_total is a global variable
4      D ← D_global
5      for i in (0, limit) do
6          d ← D.pop()
7          for p ∈ P_non_protected do
8              for δ ∈ {−1, +1} do
9                  d′ ← d;  d′[p] ← d[p] + δ
10                 if d′[p] ∈ Dom(p) then continue
11                 if d′ ∈ D_total then continue
12                 if eval_disc(d′) then
13                     // dynamic update of seed data D
14                     D.push(d′)
15                 D_total ← D_total ∪ {d′}
16     return D
```

2) Dynamic update of seed data : Naively, the number of local iterations would be the number of non-protected attributes times 2. In our running example, this iteration limit would be 24 (=12 x 2). However, this would severely limit the search space. Furthermore, in AEQUITAS, users can specify a local iteration limit (by variable *limit* in Algorithm 1), typically 1000 or 2000. These values are much higher than the naive value of 24 stated above. This suggests we need to consider cases where a user-specified local iteration limit is much higher than the naive number of local iterations.

KOSEI is designed to dynamically update the discriminatory data (called *seed data*) that the local search works on to bridge this gap. Although the local search of KOSEI starts working on discriminatory data passed by the global search as seed data, on detecting a new discriminatory data item, it appends that data item to the seed data so that the newly-detected data item can also be an object of the local search. KOSEI searches vicinities of discriminatory data detected in the global search and those of newly detected discriminatory data during the local search. The local search thus terminates when it reaches the iteration limit specified by users or when there is no more seed data.

C. Local search algorithm of KOSEI

Algorithm 3 shows the local search algorithm of KOSEI, which incorporates the two key mechanisms.

Note first that since the proposed local search offers an alternative to that of AEQUITAS, its input and output are designed the same as those of AEQUITAS's local search (in Algorithm 1).

The algorithm sets seed data D with the discriminatory data D_{global} (to be passed by the global search) at line 4. It next iterates the following procedure for the number of times specified by '*limit*': Dequeueing the first data item from the seed dataset D, it applies the perturbation (as explained in Section III-B1). That is, for each attribute of the dequeued data item, it perturbs the value of the attribute by '$-1/+1$'. In doing so, the perturbed data item is checked to see if it is valid (line 10) and if it has been evaluated previously (line 11, where D_{total} remembers the list of previously evaluated data). Next, the data item that passed these checks is evaluated for whether it is discriminatory or not (line 12). If it is evaluated discriminatory, it is appended to the list of seed data (line 14). Finally, the evaluated data (regardless of whether it is discriminatory or not) is added to D_{total} (line 15).

D. Advantages of KOSEI

KOSEI improves the limitations of the local search algorithm of AEQUITAS, discussed in Section III-A.

The first improvement is on the limitation that AEQUITAS's local search scans a search space, where there may be little discriminatory data (as stated in Section III-B1). The local search of KOSEI searches all the neighbors of a given data item, where all the neighbors of a data item mean all the perturbed data of a data item defined in Definition 2. Note, meanwhile, the local search of AEQUITAS does not necessarily work in this way according to its probabilistic search. This strategy of KOSEI aims to search data that are more likely to be discriminatory. The dynamic updating of seed data of KOSEI also can contribute to this aspect. Recall the two data d_1 and d_2 in Section III-A, around which there are respectively 10 and 1000 discriminatory data. In this situation, KOSEI may spend more search resources on d_2 than d_1 since its dynamic updating mechanism updates the seed data with more discriminatory data around d_2. KOSEI is also guaranteed to search the neighbors of discriminatory data only, which is not guaranteed in AEQUITAS.

The second is the limitation of wasting search resources to evaluate duplicated data. The local search of KOSEI carefully avoids evaluating duplicated data (as clarified in line 11 in Algorithm 3). Moreover, KOSEI is implemented without using the basin-hopping optimization function, which is heavily used in the AEQUITAS implementation. The optimization function's implementation demands a large amount of execution time since it evaluates many data evaluated in previous iterations, resulting in the inefficiency of detecting discriminatory data.

IV. EVALUATION

We conducted experiments to evaluate KOSEI. For the evaluation, we pose the two research questions (RQs).

- RQ1 Does KOSEI find more discriminatory data than AEQUITAS, at a reasonable execution cost?
- RQ2 Does KOSEI generate test cases likely to be discriminatory and avoid duplicated data, better than AEQUITAS?

RQ1 is the main RQ since the goal of this paper is to improve AEQUITAS. RQ2 investigates if and how much our improvement on local search improves the limitations of AEQUITAS, as we discussed in Section III-A.

TABLE I
COMPARISON OF KOSEI AND AEQUITAS (RQ1)

	Dataset	Classifier	#Discriminatory data			Time (s)			#Test cases		Precision (%)		Duplicated evaluation (%)
			Aequitas	KOSEI	Pct (%)	Aequitas	KOSEI	Pct (%)	Aequitas	KOSEI	Aequitas	KOSEI	Aequitas
1	Census Income	DT	2,147.5	**25,434.9**	1,184.4	57.6	**5.8**	10.0	26,970.5	35,205.3	8.1	**71.9**	95.01
2		MLPC	8,110.2	**369,629.5**	4,557.6	6,105.9	**609.4**	10.0	26,279	515,406	33.6	**71.7**	99.76
3		RF	7,506.9	**231,275.2**	3,080.8	4,465.1	**357.9**	8.0	48,838.8	333,604.5	16.6	**69.4**	99.06
4	Statlog	DT	2,135.6	**34,031.9**	1,593.6	109.4	**9.9**	9.0	23,986.9	42,405.2	9.2	**80.3**	97.75
5		MLPC	1,498.2	**101,092.7**	6,747.6	1,371.6	**52.3**	3.8	17,522.9	316,411	9.0	**32.0**	99.84
6		RF	8,727.8	**379,656.3**	4,350.0	14,442.0	**820.8**	5.7	25,202.4	707,607.1	34.6	**53.6**	99.85
7	Bank Marketing	DT	8,374.2	**206,718.3**	2,468.5	440.2	**36.5**	8.3	79,605.3	276,798.6	11.8	**74.7**	98.68
8		MLPC	1,654.8	**187,076.4**	11,305.1	746.9	**53.8**	7.2	8,001.3	302,790.9	21.4	**61.8**	99.88
9		RF	7,481.0	**783,700.6**	10,475.9	18,380.9	**1,010.1**	5.5	16,273.9	1,209,985.7	46.1	**64.8**	99.94
	Average				5,084.8			7.5			21.2	64.5	98.86

A. Experimental environment and settings

We implemented KOSEI with Python 2.7.18, extending AEQUITAS and using the scikit-learn library [8]. The code for KOSEI is available at: "https://github.com/sskeiouk/KOSEI".

For a fair comparison, we use the same settings used in [2] [6], as follows: three datasets (Census Income[3], Statlog[4], Bank Marketing[5]), three classifiers (Decision Tree (DT), MLPC, Random Forest (RF)), and protected attributes ('gender' for Census Income and Statlog, and 'age' for Bank Marketing).

For AEQUITAS, we use the fully-directed search variant since it is shown that the variant performs best among the three variants [2]. The iteration limits for global and local search (i. e., global and local iteration limit) are set to 2000, which are also the settings used in [2]. We ran ten executions for each configuration of experiments and took their average. All experiments were executed on a laptop machine with Apple M1, 16GB of RAM, running macOS Big Sur 11.4.

B. [RQ1] Does KOSEI find more discriminatory data than AEQUITAS, at a reasonable execution cost?

Table I shows the results of experiments to compare KOSEI and AEQUITAS. The rows represent configurations of datasets and classifiers. The columns represent the number of detected discriminatory data ('#Discriminatory data') and execution time ('Time'). The result shows that KOSEI's performance was orders of magnitude better than AEQUITAS. KOSEI detects more discriminatory data than AEQUITAS, by 5084.8% on average, for all nine configurations, and up to 11305.1% (for the eighth configuration).

The results also show that KOSEI requires less execution time than AEQUITAS. KOSEI runs faster than AEQUITAS, by 13.3 times on average, and KOSEI is faster by up to 26.3 times (for the fifth configuration).

Answer for RQ1

Yes. The discrimination detecting ability of KOSEI is 5084.8% stronger than AEQUITAS. Also, KOSEI runs 13.3 times faster than AEQUITAS.

[3]https://archive.ics.uci.edu/ml/datasets/adult

[4]https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[5]https://archive.ics.uci.edu/ml/datasets/bank+marketing

C. [RQ2] Does KOSEI generate test cases likely to be discriminatory and avoid duplicated data, better than AEQUITAS?

We measure the number of test cases generated by KOSEI and AEQUITAS and calculate hit ratios (i. e., precisions) of discriminatory data over generated test cases. The '#Test cases' and 'precision (%)' columns in Table I respectively show them. As stated in [3], [4], AEQUITAS generates and evaluates many duplicated data, which may cause inefficiency of the technique. Therefore, we measured the ratio of duplicated data over all the data evaluated in AEQUITAS, shown in the 'Duplicated evaluation (%)' column.

The results confirm that the precisions of KOSEI are higher than those of AEQUITAS by 3.04 (= 64.5/21.2) times on average. From the 'Duplicated evaluation (%)' column, we also observe that 98.8 % of evaluated data are duplicated. Note that KOSEI avoided evaluations of duplicated data due to the algorithm design (line 11 in Algorithm 3).

Answer for RQ2

Yes. KOSEI generates test cases more likely to be discriminatory than AEQUITAS, by 3.04 times on average. In addition, in AEQUITAS, 98.8 % of evaluated data are duplicated, while in KOSEI, all duplicated evaluations are avoided.

V. THREATS TO VALIDITY

This section discusses the main validity threats of our study.

a) Datasets, classifiers, and protected attributes used in experiments: Our evaluation experiments use the same settings used in [2] [6]: three datasets (Census, Statlog, and Bank), three classifiers (DT, MLPC, RF), and one protected attribute ('gender', 'age'). As with any experiments, the number of configurations is a threat, so additional experiments with more datasets, classifiers, and different protected attributes (e.g., 'Race') would strengthen the generalization. Note, however, that it is not easy to consider all configurations in experiments since the number of configurations increases exponentially due to combinations. Experiments in other fairness testing studies thus also pick several datasets, classifiers, and protected attributes instead of thorough configurations.

b) Comparison with other techniques: Our study evaluates KOSEI by comparing it against only AEQUITAS; that is, we did not compare it with other black-box individual fairness testing techniques, such as SG [4] and VBT [5]. The focus of the study is on improving AEQUITAS, and thus evaluation experiments were designed accordingly. Also, fair comparison with SG and VBT is not easy since their implementation details are not equivalent; e. g., KOSEI (based on AEQUITAS) uses Scikit-Learn, while VBT and SG use TensorFlow for ML library.

c) Construct validity: We implemented KOSEI by extending the AEQUITAS code, obtained from "https://github.com/sakshiudeshi/Aequitas". Despite our best efforts to pursue the code quality of KOSEI, it cannot be guaranteed that the code is free of bugs, as always. Therefore, we make the KOSEI code available online so that anyone can inspect its validity.

VI. Related work

Galhotra et al. [1] developed an individual fairness testing technique based on this abstract concept of individual fairness introduced by Dwork et al. [9]. They realize it by treating two individuals as similar if they are identical except for protected attributes. This simple treatment of individual similarity has been widely accepted, as most studies on individual fairness testing (discussed below) are based on this concept of individual similarity. This work also proposed algorithms for the individual fairness testing, called THEMIS, which are basically based on simple random testing.

Aggarwal et al. [4] proposed an individual fairness testing technique, called SG, that uses a symbolic execution technique [10]. Symbolic execution was initially developed for program analysis to systematically search the input space in order to cover input space efficiently. SG applies this technique to individual fairness testing to gain its efficacy. It is also noteworthy that SG also structures its algorithm with the global and local search phases, inspired by AEQUITAS.

Morales et al. [6] proposed an individual fairness testing technique named CGFT, which focused on the global search of AEQUITAS. While AEQUITAS used random testing in the global search, CGFT proposed using combinatorial testing (CT) [11], which has a diverse sampling ability, resulting in an improvement in the discrimination detecting ability.

Sharma and Wehrheim [5] proposed Verification-Based Testing (VBT). Its key idea is to detect discriminatory data by encoding the property of individual fairness and the approximated ML classifier under test into a logical formula and solving the encoded constraint with a constraint solver.

While the techniques mentioned above (i. e., [1], [2], [4]–[6]) are all featured with a black-box testing approach, Zhang et al. [3] proposed a white-box approach to individual fairness testing. Specifically, it targets ML classifiers based on Deep Neural Networks (DNN). Their algorithm, called ADF, is inspired by a gradient technique to detect adversarial examples. However, this technique differs from ours in that it is only applicable to DNN based ML classifiers.

VII. Conclusion and Future Work

This paper proposed a black-box individual fairness testing technique called KOSEI. KOSEI is based on AEQUITAS and is realized by improving the local search of AEQUITAS based on its limitations identified by our insight. Experiments show that the performance gain of KOSEI compared with AEQUITAS is orders of magnitude; KOSEI, on average, detects 50.8 times more discriminations than AEQUITAS. Experiments also show that the performance gain is due to our technical improvement based on our insight.

There are many directions to extend this work. The first is to extend experiments with more datasets, more classifiers, and different kinds of protected attributes (e. g., race) for further generalization. The second direction is to evaluate KOSEI, comparing it with other techniques, such as SG [4] and VBT [5]. Another direction is to evaluate the use of detected discriminatory data for re-training classifiers to improve its fairness, as attempted in [2]. Finally, we also plan to combine the improvement of the local search of KOSEI with other techniques, such as with CGFT [6], which improves the global search of AEQUITAS.

Acknowledgement

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of ESEC/FSE'17*, 2017, pp. 498–510.
[2] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of ASE'18*, 2018, pp. 98–108.
[3] P. Zhang, J. Wang, J. Sun, G. Dong, X. Wang, X. Wang, J. S. Dong, and T. Dai, "White-box fairness testing through adversarial sampling," in *Proceedings of ICSE'20*, 2020, pp. 949–960.
[4] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of ESEC/SIGSOFT FSE'19*, 2019, pp. 625–635.
[5] A. Sharma and H. Wehrheim, "Automatic fairness testing of machine learning models," in *Proceedings of ICTSS'20*, 2020, pp. 255–271.
[6] D. P. Morales, T. Kitamura, and S. Takada, "Coverage-guided fairness testing," in *Proceesing of ICIS'21*, 2021, pp. 183–199.
[7] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[9] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of ITCS'12*, 2012, pp. 214–226.
[10] K. Sen, D. Marinov, and G. Agha, "CUTE: a concolic unit testing engine for C," in *Proceedings of FSE'05*, 2005, pp. 263–272.
[11] R. Kuhn and R. Kacker, *Introduction to Combinatorial Testing*. Chapman & HallCRC, 2013.

Data Driven Testing for Context Aware Apps

Ryan Michaels
St Edward's University
rmichael@stedwards.edu

Shraddha Piparia
University of North Texas
ShraddhaPiparia@my.unt.edu

David Adamo
Block, Inc.
dadamo@squareup.com

Renee Bryce
University of North Texas
Renee.Bryce@unt.edu

***Abstract*—Context driven environments are growing in popularity. Mobile applications, Internet of Things devices, autonomous vehicles, and future technologies respond to context events in their environments. This work uses a set of context events from real users to guide the generation of context driven test cases. Context event sequences are obtained by applying Conditional Random Fields (CRF). Test suites are then constructed by interleaving the context event sequences with GUI events. The choice of context event is made based on transitions obtained from the CRF. Results of the empirical studies show that techniques that incorporate context events provide better code coverage than NoContext for the subject applications. A heuristic technique introduced in this work, ISFreqOne, yields 4x better coverage than NoContext, 0.06x better coverage than Random Start Context, 0.05x better coverage than Iterative Start Context, which are control context generation techniques, and 0.04x better coverage than ISFreqTwo, another heuristic introduced in this work.**

***Index Terms*—Android Testing, Context events, GUI events, Software Testing, Test Suite Generation**

I. Introduction

Many applications allow streams of context and user events to influence their behavior. We see examples in the domains of autonomous vehicles, Internet of Things (IoT), and mobile devices. For instance, if a user clicks a button, an app that responds by accessing data over the Internet to update the user's view will respond differently the device's context changes from WiFi to airplane mode. Additional examples of context events include changes in battery levels, the device's physical location, sound output to speakers or headphones, and changes to screen orientation. The work in this paper generates context aware test suites with a strategy based on a real-world data set of context events [1].

In the remainder of the paper: Section II summarizes related work; Section III describes the event sequence model; Section IV covers the data driven test generation strategy; Section V describes experiments; Section VI shows results and discussion; Section VII shares threats to validity; and Section VIII gives conclusions.

II. Related Work

GUI testing: GUI testing is an important task that many tools support. Examples include Monkey [2], Dynodroid [3], and Autodroid [4]. Most tools generate test cases without consideration of context events and their interactions with the application under test (AUT). Tests are often generated with respect to an initial setup of context variables that do not change during the testing. This may result in insufficient exploration of application states and code.

Monkey [2] offers fast and replayable test cases in the form of random clicks and swipes. Monkey does not interact explicitly with on-screen elements such as buttons and text fields, but clicks on events at specific screen coordinates [2].

Dynodroid [3] is an online testing tool that is responsive to application changes when it generates a "next event". It considers both system and user generated events. In an extensive study, Dynodrod generated 50 open source applications and outperformed Monkey in terms of code coverage.

Tema [5], [6] is an online GUI testing framework that utilizes models of application behavior informed by user data to generate abstract tests and are independent of device platforms. The models identify application state from abstract user actions. In its final step, Tema translates the abstract test cases into test cases by mapping of actions for specific devices and applications.

Context-aware GUI testing: Testing that uses both context and GUI events may increase fault detection for context driven apps [3], [7]–[10]. Dynodroid [3] is one tool that considers context events. While generating context events, Dynodroid does not offer a guarantee of combinatorial coverage of context and GUI event intersections will occur.

Adamsen et al. [11], Majchrzak et al. [12], and Song et al. [10] each propose context-sensitive mobile testing approaches. The approaches execute test suites under differing context environments. The change in context can change a valid test into an invalid one, such as a video streaming app attempting to execute without WiFi. While their approaches cover multiple context environments, there are opportunities to improve with more cost-effective strategies.

Amalfitano et al. [7] use context and GUI event combinations into test case generation. They consider a small set of GUI and context events and interleave them during test generation. The work demonstrates benefits of testing with both context and GUI events.

Griebe et al. [8] use a model-based approach where testers generate an annoted UML diagram to describe GUI behavior and context parameters of the AUT. The authors later expand this approach to incorporate sensor input [13]. Using the UI sensing tool Calabash-Android, they generate sensor values into the test cases [14]. They further provide parsing of natural language expressions, such as "I invert the phone" to generate additional test data.

DOI reference number: 10.18293/SEKE2022-125

CAIIPA [15] is a cloud based testing service that supports context aware apps. Their results demonstrate that using real-world context events during test generation improves performance fault and crash detection up to 11x better than Monkey. CAIIPA utilizes real-world context data. However, it is limited to hardware options such as WiFi and sensor settings. It cannot detect context such as screen orientation. AppDoctor [16] is another a cloud-based automation testing tool that injects events such as changes to network states, device storage, GUI gestures, and more during execution.

MoTiF [17] identifies and replicates context sensitive crashes for Android apps. Future work may extend this to not only reproduce crashes but to fully identify information of crash patterns across applications.

MobiCoMonkey [18] extends Monkey [2] by itnerleaving context events at random or as predefined by testers. This tool could be enhanced in the future with systematic interleaving of context and GUI events.

Conditional Random Fields (CRFs): Hidden Markov Models(HMMs) [19] are popular probabilistic sequence models [20]. However, HMMs suffer difficulty in modeling arbitrary, dependent features of input sequences. To overcome this drawback, we apply conditional random fields (CRF) [21] to testing context-aware systems. CRF sequence models are discriminative in nature, which allows for maximization of conditional likelihood.

III. Event Sequence Model

Our event sequence model is built on Autodroid [22] which comes with a test builder, abstraction manager to identify GUI events available in the AUT at different states, an event selector, and the event executor. We add context events to enable dynamic context-GUI testing for context aware applications.

Let us define GUI and context actions and events.

Definition 1. We define individual context events as a 2-tuple (c, a); c is a context variable with a as the assigned context action.

Screen orientation	WiFi	Battery	AC power
portrait	connected	Ok	connected
landscape	disconnected	low	disconnected
-	-	high	-

TABLE I
EXAMPLE CONTEXT VARIABLES AND POSSIBLE VALUES

Table I shows four context variables (screen orientation, WiFi, battery status, AC power) that may be set to any of the respective values shown in the table. A context event is then defined as one of these variables with an assigned value, i.e. c= ScreenOrientation = *landscape*, WiFi = c*disconnected*, Battery = *low*, Power = *disconnected*.

Definition 2. Action: We define an *action* as a user interaction with the application *or* a system level call with a target. Each action consists of a *target*, *type*, and *value*. There are two types of actions:

- **GUI action:** A user executes an action using GUI widgets, e.g. a click on a on-screen widget or filling in a text-field.
- **Context action:** The mobile operating system executes a system action, e.g. change in screen orientation.

Sometimes an action may require a *value*. For example, a text box may require users to type in a value. We consider two (or more) actions to be equivalent if they have the same target and type.

Definition 3. Event: We define an *Event* consists of a sequence of actions with pre and post conditions. A GUI event causes a change in the GUI state after its execution. A context event has one or more context actions and may or may not cause a change in the GUI state. An event is a complete event if it is executed and its post condition is known, otherwise it is a partial event.

Definition 4. Test Case: We define a test case as a sequence of events. Each test case has a unique id, a set of events, and a length. Table II shows a test case of length two containing one context event and one GUI event.

ID	tc001
Event 1 (GUI event - launch)	**event**: **precondition**: **activity name**: null **state id**: null **postcondition**: **activity name**: MainActivity **state id**: 07f24 **actions**: **type**: launch **value**: null **target**: **selector**: system **selector value**: app **type**: app **description**: launch
Event 2 (Context event - Power disconnected)	**event**: **precondition**: **activity name**: MainActivity **state id**: 07f24 **postcondition**: **activity name**: MainActivity **stateId**: 5d158 **actions**: **type**: power_disconnected **value**: null **target**: **selector**: power **selector value**: disconnected **type**: context **description**: power disconnected

TABLE II
A TEST CASE WITH ONE CONTEXT EVENT AND ONE GUI EVENT

IV. Data Driven Test Generation Strategy

Data collection and CRF modeling. To generate the CRF model, we observed everyday smartphone use from 58 university students for the duration of one month, via a monitoring app [1]. The app was configured to listen to 144 broadcasts which occurred 16,257,795 times during the one month period. [1]. This data fed into the construction of CRFs using

Fig. 1. Pseudocode for data-driven test generation algorithm based on Autodroid framework [23]

Inputs: Android app. package (*AUT*), combinatorial context model (*M*), context event sequence (*C*) context event frequency (*p*)

Outputs: Test suite (T)

1: $C_{all} \leftarrow$ *generateContextsFromCoveringArray*(M)
2: $T \leftarrow \phi$ ▷ test suite
3: **repeat**
4: $T_i \leftarrow \phi$ ▷ test case
5: $e_{context} \leftarrow$ *selectInitialContext*(C_{all})
6: $T_i \leftarrow T_i \cup e_{context}$
7: install AUT and execute launch event, e_{launch}
8: $T_i \leftarrow T_i \cup e_{launch}$
9: $s_{curr} \leftarrow$ initial GUI state after app launch
10: **while** *termination condition is false* **do**
11: $E_{all} \leftarrow$ *getGuiEvents*(s_{curr})
12: $e_{sel} \leftarrow$ *selectGuiOrContextEvent*(s_{curr}, E_{all}, C, p)
13: execute selected event, e_{sel}
14: $T_i \leftarrow T_i \cup e_{sel}$
15: $s_{curr} \leftarrow$ current GUI state
16: **end while**
17: *tearDownTestCase()*
18: $T \leftarrow T \cup \{T_i\}$
19: **until** *completion condition is false*

142,138 instances for training data and 28,663 instances of context events for test data.

The CRF is the graph obtained after considering the top likely transitions. We only consider the dependency between a predefined subset of events and remove all other external dependencies. Our aim is to find a context sequence which represents these dependencies without self-loops. We then select the transitions with the highest weight.

Test generation algorithm. The algorithm in Figure 1 uses context event sequences derived from a CRF to generate a sequence of context and GUI events, and then saves the generated sequence as a test case. The test generation algorithm consists of the following inputs:

- A compiled Android application
- Context event sequence, *C*, obtained from the CRF
- An integer value, *p*, that determines how often a context event will be added to a test case

The following discussion walks through the algorithm:

Step 1: Generate context covering array. Line 1 uses the combinatorial context model provided as input to generate a covering array. Each entry in the covering array represents a possible starting context for one or more test cases.

Step 2: Test case setup. Lines 4-9 represent the test setup for each test case. Line 4 initializes the test case as an empty event sequence. At the beginning of each test case, line 5 chooses an initial context from the covering array generated in step 1 and applies the chosen context to the execution environment. Line 6 adds the starting context event to the test case. Line 7 installs the AUT and launches it in the starting context. Line 8 adds the launch event to the test case. Line 9 retrieves the initial GUI state of the AUT after launch.

Step 3: Select and execute an event. Line 11 identifies all GUI events that are available and executable in the current GUI state. Lines 12-14 use the current GUI state, the context event sequence C derived from the CRF, and the specified context event frequency p to choose which GUI event or context event to execute.

The algorithm repeats steps 1 to 3 until a specified test case termination condition holds true. Context events are added to the test case at the predefined frequency p until the test case ends or until all events in the context event sequence, C, have been executed.

Step 4: Test case teardown. After the algorithm executes the last event in each test case, line 17 resets the test environment to allow tests to run independent of each other. The algorithm repeats steps 1 to 4 to generate multiple test cases containing a mix of context events and GUI events until a specified test suite completion condition holds true.

V. Experimental Setup

For our experiments we consider four applications, described in Table III. The applications have between 1,215 - 15,062 lines of code, 197-1134 methods, and 46-209 classes. Each application has over 1,000 downloads at the time of our experiment. The application's apk files are downloaded from F-Droid [24].

App Name	Installs	Vers.	Lines	Methods	Classes
Diode	10k+	1.3.2.2	7,933	1134	209
Your Local Weather	5k+	5.6.4	15,062	499	114
MovieDB	1k+	2.1.1	2,719	319	81
Abcore	1k+	0.77	1,215	197	46

TABLE III
Characteristics of the apps under test

A. Experimental Setup

The experiments use the Android 10.0 Pixel emulator (API 29) to generate ten test suites of two hour duration for each technique and application in the study. A two second delay occurs between event executions and there is a .05 probability to terminate a test case. We instrument subject applications with JaCoCo [25] to measure coverage.

B. Variables and Measures

Independent Variable. The independent variable of our experiments is our test generation technique. We consider three control techniques and two heuristic techniques.

Control techniques:

- **NoContext** generates a test suite of GUI events with only an initial set of context variables that do not change during testing. *NoContext* construct test suites

using c = *ScreenOrientation=Portrait WiFi=connected, Battery=OK, AC Power=connected*}.

- **RandomStartContext (RSContext)** starts each test case by selecting a start context at random from a context covering array and then makes random selections of only GUI events.
- **IterativeStartContext (ISContext)** starts each test case by selecting a start context in a round-robin fashion from a context covering array and then makes a random selection of only GUI events.

Heuristic techniques:

- **IterativeStartFreqOne (ISFreqOne)** iterates through the context covering array to set a starting context and then uses context sequences obtained from the CRF to interleave context events with GUI events at an *interval of one* until all events in the context sequence have been executed.
- **IterativeStartFreqTwo (ISFreqTwo)** iterates through the context covering array to set a starting context and then uses context sequences obtained from the CRF to interleave context events with GUI events at an *interval of two* until all events in the context sequence have been executed.

The heuristic techniques, *ISFreqOne* and *ISFreqTwo*, use context event sequences derived from a CRF that includes only events for internet connectivity, power connection, battery level, and screen orientation changes.

Dependent Variables. We assess our research questions using code coverage:

- **Line coverage** measures the number of lines of code executed by the test suite relative to the total number of statements in the AUT.
- **Method coverage** counts the number of methods executed by the test suite relative to the total number of methods in the AUT.
- **Class coverage** counts the number of classes executed by the test suite relative to the total number of classes in the AUT.

C. Research Questions

The experiments examine two research questions:

RQ1: Do ISFreqOne and ISFreqTwo increase line, method, and class coverage in comparison to NoContext, RSContext, and ISContext?

RQ2: Which of the two heuristic techniques provide the greatest coverage of lines, methods, and classes in the test applications?

VI. Results and Analysis

RQ1 Results: Tables IV, V, and VI show the average results for line, method, and class coverage for each technique and app. We calculate the ratio of our techniques to the controls by dividing average values of both heuristics (ISFreqOne, ISFreqTwo) by control techniques (NoContext, RSContext, and ISContext). The ISFreqOne technique shows an improvement of approximately four times (312%) line coverage, three times (191%) method coverage, and 3.5 times (251%) class coverage when compared to NoContext for the application AbCore. IsFreqOne shows an improvement of 1.2 times (17%) line coverage, 1.2 times (21%) method coverage, and 1.2 times (18%) class coverage for ISFreqOne when compared to NoContext for the application MovieDB. ISFreqOne shows about 4% increase in line, method, and class coverage when compared to NoContext for the application Diode. ISFreqOne does not result in improvements for Your Local Weather relative to NoContext. For the Your Local Weather application, method coverage values for ISFreqOne and NoContext are similar although the class coverage is less for ISFreqOne when compared to NoContext. Your Local Weather has low overall code coverage. Our tool could not explore the application much because it needs a location to be entered or selected from the map which hindered exploration. The average improvement of ISFreqTwo over NoContext is 1.8 times line coverage, 1.5 times method coverage, and 1.67 times class coverage across all four applications.

	NoContext	RSContext	ISContext	ISFreqOne	ISFreqTwo
Abcore	15.83	62.87	63.95	65.15	62.83
MovieDB	40.65	45.22	45.45	47.5	47.205
YourLocalWeather	9.05	9.09	9.04	9.05	9.03
Diode	32.44	32.89	33.33	33.69	33.34

TABLE IV
AVERAGE LINE COVERAGE

	NoContext	RSContext	ISContext	ISFreqOne	ISFreqTwo
Abcore	25.38	71.72	72.88	73.81	71.28
MovieDB	47.37	54.12	54.55	57.365	55.175
YourLocalWeather	15.32	15.33	15.15	15.18	15.15
Diode	43.81	44.08	45.37	45.50	44.32

TABLE V
AVERAGE METHOD COVERAGE

On average, ISFreqOne offers improvement of 3.03% line coverage, 3.35% method coverage in comparison to RSContext across all four applications for ISFreqOne over RSContext. There is an average improvement of 2.2% line coverage, 0.6% method coverage, and 1.1% class coverage for ISFreqTwo over NoContext. Across all the applications, there is 2.2% line coverage, 2.14 method coverage, and 0.17% class coverage improvement in ISFreqOne over ISContext. We observe an improvement of 1.1% line coverage and 0.92% class coverage in ISFreqTwo over ISContext across all four applications. The method coverage, on average, did not show improvement for ISFreqTwo over ISContext. The noted improvements in code coverage indicate that for context-aware mobile applications, the presence or absence of context events in test suites has noticeable effects on the behavior of applications and the ability of test suites to adequately explore application functionality. The results also suggest that context events in test suites are useful not just at the beginning of test cases but also mixed in with GUI events at different intervals within each test case.

RQ2 Results: We compare results obtained from both heuristic techniques: ISFreqOne and ISFreqTwo. ISFreqOne outperforms ISFreqTwo across two subject applications. For Movie DB, there is an improvement of 0.62% line coverage

	NoContext	RSContext	ISContext	ISFreqOne	ISFreqTwo
Abcore	21.74	76.30	77.33	76.24	76.39
MovieDB	51.36	59.88	59.45	60.49	61.11
YourLocalWeather	27.93	27.93	25.88	25.88	25.88
Diode	38.42	39.27	40.08	40.10	39.98

TABLE VI
AVERAGE CLASS COVERAGE

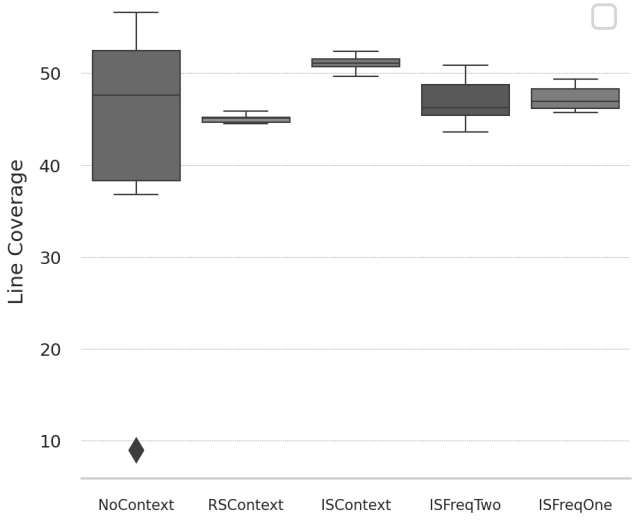


Fig. 2. Application *Movie DB*: Box plot of NoContext, RSContext, ISContext, ISFreqTwo, and ISFreqOne

and 3.9% method coverage. Class coverage is 1.03% higher for ISFreqTwo compared to ISFreqOne. The application AbCore shows an improvement of 3.7% line coverage and 3.5% method coverage for ISFreqOne over ISFreqTwo. Class coverage is 2.1% higher for ISFreqTwo over ISFreqOne. Diode application shows an improvement of 1.18% line coverage, 3.17% method coverage, and 0.42% class coverage for ISFreqOne over ISFreqTwo. The application Your Local Weather performs similarly for both of these techniques. On an average, there is an improvement of 1.15% line coverage and 3.76% method coverage for ISFreqOne over ISFreqTwo with 0.18% less class coverage. These results indicate that inserting context variables at an interval of one GUI events shows more coverage than inserting at two GUI events. This could be due to the fact that the Android applications are small in size. Figures 2 - 5 show the box plot the line coverage of all five techniques for Movie DB, AbCore, Diode, and Your Local Weather respectively. The standard deviation is less for ISFreqOne when compared to ISFreqTwo for all applications, indicating ISFreqOne is the more reliable generation heuristic on the AUTs in this study.

VII. Threats to Validity

The subject applications used for the experiments have different characteristics which may affect the performance of our techniques. Adding more subject applications can help to better generalize the results. We tried to minimize this threat by choosing applications of varying sizes from

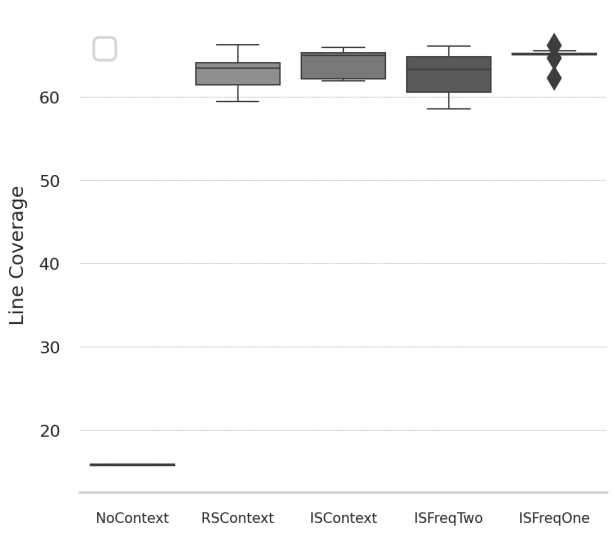


Fig. 3. Application *AbCore*: Box plot of NoContext, RSContext, ISContext, ISFreqTwo, and ISFreqOne

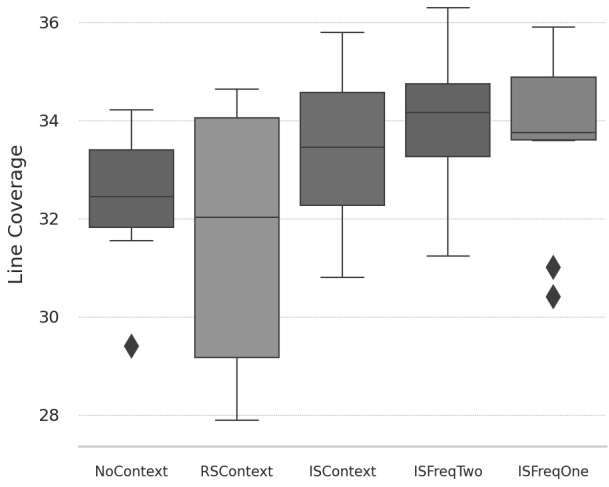


Fig. 4. Application *Diode*: Box plot of NoContext, RSContext, ISContext, ISFreqTwo, and ISFreqOne

different domains. The set of context events used in this experiment are limited in number and do not cover the totality of context events in Android. We minimized this threat by focusing on events that are reliably accessible in the emulator. We specifically excluded sensor events due to the sheer amount of data produced by sensors and the battery-intensive operations required to collect a constant stream of sensor values which helps in minimizing this threat. Inclusion of more context events may change results. For instance, we did not consider internet changes in this work. This was because the values in our data contain several internet connections such as HSPA_CONNECTION, HSDPA_CONNECTION, LTE_CONNECTION, etc. We hope that future advancements in emulators or inexpensive device

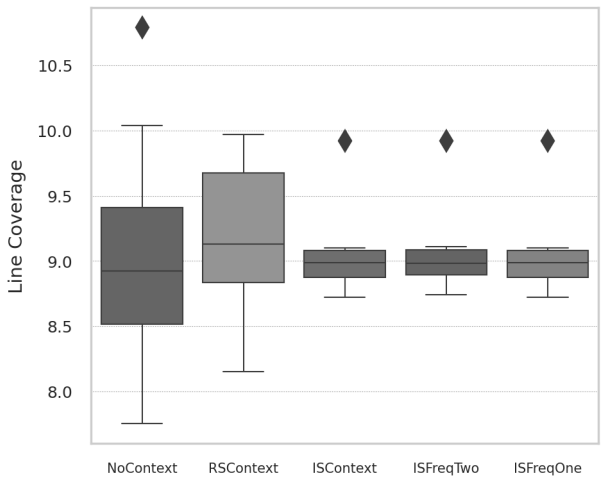


Fig. 5. Application *Your Local Weather*: Box plot of NoContext, RSContext, ISContext, ISFreqTwo, and ISFreqOne

farms will allow such expansions.

VIII. Conclusions

This work develops and studies algorithms that systematically generate test suites comprised of context events and GUI events. In particular, these techniques are guided by a real world data set of context data from 58 users. The choice of context events is made based on transitions obtained from CRF. We analyze how often context change should occur by providing an interval as a parameter to the tool. We observe that the techniques that incorporate context events performed better than NoContext among all four subject applications. The heuristic technique ISFreqOne yields four times better coverage than NoContext, 0.06 times better coverage than RSContext, 0.05 times better coverage than ISContext, and 0.04 times better than ISFreqTwo strategies. ISFreqOne also has a lower standard deviation between runs. The initial context strategies RSContext and ISContext performed better than NoContext by up to a factor of four indicating the importance of context while testing Android applications.

Future work will extend this study to examine larger context data sets and apply the techniques to more mobile applications. This work also serves as a foundation for future work that extends our techniques to other domains such as Internet of Things (IoT), wearable technologies, and autonomous vehicles. Future work will further examine fault finding effectiveness of context-aware test generation.

References

[1] S. Piparia, M. K. Khan, and R. Bryce, "Discovery of real world context event patterns for smartphone devices using conditional random fields," in *ITNG 2021 18th International Conference on Information Technology-New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2021, pp. 221–227.

[2] Google Inc, "Ui/application exerciser monkey." [Online]. Available: http://developer.android.com/tools/help/monkey.html Accessed 26 Dec 2021.

[3] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An input generation system for android apps," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM, 2013, pp. 224–234.

[4] D. Amalfitano, N. Amatucci, A. R. Fasolino, and P. Tramontana, "A Conceptual Framework for the Comparison of Fully Automated GUI Testing Techniques," in *International Conference on Automated Software Engineering Workshop (ASEW)*, 2015, pp. 50–57.

[5] A. Jaaskelainen, M. Katara, A. Kervinen, M. Maunumaa, T. Paakkonen, T. Takala, and H. Virtanen, "Automatic gui test generation for smartphone applications-an evaluation," in *International Conference on Software Engineering-Companion Volume*. IEEE, 2009, pp. 112–122.

[6] A. Nieminen, A. Jaaskelainen, H. Virtanen, and M. Katara, "A comparison of test generation algorithms for testing application interactions," in *Intl. Conference on Quality Software*. IEEE, 2011, pp. 131–140.

[7] D. Amalfitano, A. R. Fasolino, P. Tramontana, and N. Amatucci, "Considering context events in event-based testing of mobile applications," in *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2013, pp. 126–133.

[8] T. Griebe and V. Gruhn, "A model-based approach to test automation for context-aware mobile applications," in *ACM Symposium on Applied Computing*. ACM, 2014, pp. 420–427.

[9] Z. Liu, X. Gao, and X. Long, "Adaptive random testing of mobile application," in *International Conference on Computer Engineering and Technology (ICCET)*, vol. 2. IEEE, 2010, pp. V2–297.

[10] K. Song, A. R. Han, S. Jeong, and S. Cha, "Generating various contexts from permissions for testing android applications," in *Software Engineering and Knowledge Engineering (SEKE)*, 2015, pp. 87–92.

[11] C. Q. Adamsen, G. Mezzetti, and A. Møller, "Systematic execution of android test suites in adverse conditions," in *International Symposium on Software Testing and Analysis*. ACM, 2015, pp. 83–93.

[12] T. A. Majchrzak and M. Schulte, "Context-dependent testing of applications for mobile devices," *Open Journal of Web Technologies (OJWT)*, vol. 2, no. 1, pp. 27–39, 2015.

[13] T. Griebe, M. Hesenius, and V. Gruhn, "Towards automated UI-tests for sensor-based mobile applications," in *Intl. Conf. on Intelligent Software Methodologies, Tools, and Techniques*. Springer, 2015, pp. 3–17.

[14] Uber, "Calabash-android," 2019, retrieved Feb 25, 2020 from https://github.com/calabash.

[15] C.-J. M. Liang, N. D. Lane, N. Brouwers, L. Zhang, B. F. Karlsson, H. Liu, Y. Liu, J. Tang, X. Shan, R. Chandra, and F. Zhao, "Caiipa: Automated large-scale mobile app testing through contextual fuzzing," in *Intl. Conf. on Mobile Computing and Networking*. New York, NY, USA: ACM, 2014, pp. 519–530.

[16] G. Hu, X. Yuan, Y. Tang, and J. Yang, "Efficiently, effectively detecting mobile app bugs with appdoctor," in *Proceedings of the Ninth European Conference on Computer Systems*, 2014, pp. 1–15.

[17] M. Gómez, R. Rouvoy, B. Adams, and L. Seinturier, "Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring," in *Proceedings of the International Conference on Mobile Software Engineering and Systems*, ser. MOBILESoft '16. New York, NY, USA: ACM, 2016, pp. 88–99.

[18] A. S. Ami, M. M. Hasan, M. R. Rahman, and K. Sakib, "Mobicomonkey - context testing of android apps," in *Intl. Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, May 2018, pp. 76–79.

[19] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[20] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," in *In Proceedings of the Third Conference on Applied Natural Language Processing*, 1992, pp. 133–140.

[21] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Foundations and Trends® in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012. [Online]. Available: http://dx.doi.org/10.1561/2200000013

[22] D. Adamo, D. Nurmuradov, S. Piparia, and R. Bryce, "Combinatorial-based event sequence testing of android applications," *Information and Software Technology*, vol. 99, pp. 98–117, 2018.

[23] S. Piparia, D. Adamo, R. Bryce, H. Do, and B. Bryant, "Combinatorial testing of context aware android applications," in *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2021, pp. 17–26.

[24] F-Droid, "F-droid: Free and open source android app repository," http://f-droid.org, 2017, (Accessed: 26-12-2021).

[25] Mountainminds GmbH, "EclEmma: JaCoCo java code coverage library," http://www.eclemma.org/jacoco/, 2017, (Accessed: 26-12-2021).

A Preliminary Study on the Explicitness of Bug Associations

Zengyang Li†, Jieling Xu†, Guangzong Cai†, Peng Liang‡, Ran Mo†
†School of Computer Science & Hubei Provincial Key Laboratory of Artificial Intelligence and Smart Learning, Central China Normal University, Wuhan, China
‡School of Computer Science, Wuhan University, Wuhan, China
zengyangli@ccnu.edu.cn, {xjling, guangzongcai}@mails.ccnu.edu.cn, liangp@whu.edu.cn, moran@ccnu.edu.cn

***Abstract*—Bugs are usually in associations with other bugs in a software system, e.g., a bug may result from another bug. However, such bug associations are implicit and usually cannot be traced without a significant amount of effort. Intuitively, if a bug association is easier to trace, the involved bugs can be fixed in a cleaner way. However, there is little evidence on the explicitness of bug associations. In this paper, we aim to evaluate the explicitness of bug associations, so as to get a basic understanding on such associations. To this end, we defined a metric to quantify the explicitness of a bug association, and conducted an empirical study on 11 non-trivial Apache open source software systems. The main findings are summarized as follows: (1) From the perspective of code change history, around 29% of bug pairs are not explicitly associated, and about 71% are explicitly associated to some extent; (2) Bugs in the association of Container have relatively strong association explicitness, while bugs in the association of Blocked or Blocker, Cloners, and Dependent have relatively weak association explicitness. These findings provide insights on software analyzability to practitioners and researchers.**

***Keywords*—bug association, association type, empirical study, open source software**

I. Introduction

Bugs are usually in association with other bugs in a software system [1], [2]. For instance, in project Apache *HBase*, bug *HBASE-21551* on memory leak was caused by bug *HBASE-20704* that did not handle file storage properly. Such associations are important to software maintenance in the sense that they can facilitate locating bugs and analyzing the impact of the bugs on the system [3], [4]. Intuitively, if a bug association is easier to trace, the involved bugs can be fixed in a cleaner way, i.e., the bugs can be solved more completely. Despite of the importance of bug associations, there is little evidence on the explicitness of bug associations from the perspective of change history of the software system. Hence, in this work, we aim to evaluate the explicitness of bug associations, so as to get a basic understanding on such associations.

To this end, we conducted an empirical study on 11 non-trivial Apache open source software (OSS) projects, which bugs are managed in JIRA, an issue tracking system deployed by Apache Software Foundation. In JIRA, for each project, a proportion of bug associations are manually labeled by practitioners, but it is not clear whether and to what extent such manually labeled associations can be reflected in the code change history (i.e., commits) of the project. The results of this question can reflect the difficulty of identifying the bug associations and the possibility of automatic identification.

Our main contributions lie in the following two aspects. (1) This work is an early attempt to explore the explicitness of bug associations from the perspective of software change history. (2) We defined a metric to quantify the explicitness of bug associations, and examined the bug association explicitness using this metric in 11 Apache OSS projects.

This work is supported by the Natural Science Foundation of Hubei Province of China under Grant No. 2021CFB577 and the Natural Science Foundation of China (NSFC) under Grant No. 62172311.
DOI reference number: 10.18293/SEKE2022-027

II. Background and Related Work

A. Background

There are different types of the association between two bugs. We collected 16 types of bug associations from JIRA[1], including: (1) Blocked, (2) Blocker, (3) ChildIssue, (4) Cloners, (5) Container, (6) Dependency, (7) Dependent, (8) Duplicate, (9) Incorporates, (10) ParentFeature, (11) Problem/Incident, (12) Reference, (13) Regression, (14) Related, (15) Required, and (16) Supercedes. The details of all the 16 bug association types are provided online[2].

B. Links between Issues

Many studies investigated various links (e.g., associations or dependencies) between issues (including bugs) and the characteristics and applications of these links. Kucuk *et al.* classified duplicate bugs based on analyzing the difference between duplicate and non-duplicate bugs [5]. Tomova *et al.* studied issue link type selection [6]. Researchers use trace links to support various development and maintenance tasks, such as impact analysis [3] and bug tracking [7]. However, our work is not to explore what kind of relationship between bugs, but to study the explicitness of the association between bugs that have been manually linked from the perspective of code change history.

C. Links between Bugs and Commits

Numerous studies investigated the links between bugs and commits on the detection and application of the links. Le *et al.*

[1]https://issues.apache.org/jira
[2]https://github.com/breezesway/BugAssociationType

created a discriminative model for predicting whether there is a link between commit messages and bug reports [8]. Li *et al.* made use of links between bugs and commits to identify bug-fixing commits and then calculated the change complexity of bug-fixing commits [9], [10]. However, these studies do not discuss the relationship between the corresponding commits and associated bugs, which is the focus of our study.

III. Study Design

To investigate the explicitness of bug associations, we conducted a preliminary case study on Apache OSS projects.

A. Research Questions

RQ1: How much explicitness do the bug associations have?
Rationale: The explicitness of bug associations helps to reveal how difficult the association is to be identified, thereby assessing the likelihood that the bug can be completely resolved to some extent. We quantify the explicitness of bug associations for each project, and study how the explicitness is distributed.
RQ2: Is there a difference on the association explicitness between different association types?
Rationale: We study whether there are significant differences between different association types, in order to understand whether the association explicitness of different association types is different. This gives researchers and developers inspiration for whether to pay different attention to different association types.

B. Case Selection

In this study, we only investigated Apache OSS projects which main programming language is Java. For selecting each case (i.e., OSS project) included in our study, we applied the following criteria: C1) Over 85% of the source code is written in Java; C2) The history of the project is more than 5 years; C3) The number of commits of code repository of the project is more than 4000; and C4) The number of bug pairs that are manually associated in JIRA is more than 100. C1 is set to ensure that the strength of the association between bugs is clearly defined. C2 and C3 are set to ensure that the selected project is non-trivial and has sufficient data. C4 was set to ensure that the final sample dataset for analysis was large enough for statistical analysis.

C. Data Collection

To answer the RQs formulated in Section III-A, we collected the data items listed in TABLE I, which also provides the mapping between the data items and the target RQ(s). All data items were collected from JIRA and GitHub.

TABLE I: Data items to be collected.

#	Name	Description	Target RQ(s)
D1	Associated bug pair	A pair of bugs in association.	RQ1
D2	Association type	The type of association between each pair of bugs.	RQ2
D3	Changed source files	The number of Java source files modified in the bug-fixing commit(s).	RQ1, RQ2

D. Data Analysis

To answer RQ1, we first defined a metric, namely Association Explicitness or AE, to quantify the explicitness of the association between two bugs. Assuming that the pair of bugs α and β are in an association manually labeled in JIRA, the set of Java source files modified in the commits for fixing α is F_α, and the set of source files modified in the commits for fixing β is F_β. The AE of the association between α and β is defined as follows:

$$AE = \frac{|F_\alpha \bigcap F_\beta|}{|F_\alpha \bigcup F_\beta|} \quad (1)$$

The AE of a bug pair falls into [0.0,1.0]. Second, we calculated the distribution of the proportion of bug pairs against total bug pairs over different intervals of AE value. We divide the interval into [0.0,0.0], (0.0,0.1], (0.1,0.2], ..., (0.9,1.0), [1.0,1.0]. Especially, we count the cases where two bugs are fixed in the same commit(s) in the case of [1.0,1.0].

To answer RQ2, taking all bug pairs of the selected projects as a whole, we ran the Mann-Whitney U tests to calculate whether there is a significant difference on AE between bug pairs of different association types.

IV. Study Results

A. Explicitness of Bug Associations (RQ1)

TABLE II shows the distribution of percentage of bug pairs over intervals of the AE value for the 11 projects. (1) The percentage of the bug pairs with $AE = 0.0$ of each project ranges from 22.5% to 41.1%. When taking all projects as a whole, there are around 29.4% bug pairs with $AE = 0.0$. This indicates that for those bug pairs, no source files are changed in the bug-fixing commits for both bugs of each bug pair. (2) Consider the AE interval of (0.0,0.5]. Taking all projects as a whole, the AE of about 52.6% of bug pairs falls into this interval. (3) Consider the AE interval of (0.5,1.0). Taking all projects as a whole, the AE of about 4.1% of bug pairs falls into this interval. (4) There are 4.6%-35.1% of bug pairs that have a perfect association explicitness, i.e., $AE = 1.0$, for each project. Taking all projects as a whole, 13.9% of the bug pairs are with $AE = 1.0$, which means that the same source files are changed in the bug-fixing commits of the two bugs of each of those bug pairs.

We further studied the bug pairs in which the two associated bugs are fixed in the same commit(s). The results are shown in TABLE III, where column *#BugPairA* denotes the number of bug pairs with $AE = 1.0$, column *#BugPairSC* denotes the number of bug pairs fixed in the same commit(s), and *%BugPairSC* denotes the percentage of *#BugPairSC* over *#BugPairA*. For each project, the *%BugPairSC* ranges from 0.0% from 68.6%. Especially, projects *Accumulo* and *Hadoop* do not have any bug pairs fixed in the same commit(s).

Summary: From the perspective of code change history, on average, 29.4% of bug pairs are not explicitly associated, while 70.6% of bug pairs are explicitly associated; furthermore, 52.6% of bug pairs have a relatively low association

TABLE II: Distribution of the percentage of bug pairs against the total bug pairs over intervals of AE for the selected projects.

Project	[0.0,0.0]	(0.0,0.1]	(0.1,0.2]	(0.2,0.3]	(0.3,0.4]	(0.4,0.5]	(0.5,0.6]	(0.6,0.7]	(0.7,0.8]	(0.8,0.9]	(0.9,1.0)	[1.0,1.0]
Accumulo	38.2	13.6	20.0	7.3	10.0	6.4	0.0	0.0	0.0	0.0	0.0	4.6
ActiveMQ	33.1	9.7	17.7	9.7	6.5	4.8	1.6	3.2	0.8	1.6	0.0	11.3
Calcite	22.5	23.9	15.2	12.3	6.5	5.8	0.0	2.2	0.0	0.0	0.0	11.6
Hadoop	23.2	10.6	16.7	8.9	12.3	9.2	1.4	2.1	0.3	0.3	0.0	15.0
HBase	29.1	14.9	17.9	7.6	8.9	7.6	0.7	2.7	1.0	0.0	0.3	9.3
Hive	25.5	12.2	17.8	7.4	9.0	12.4	0.7	2.1	0.9	0.5	0.0	11.6
Jackrabbit Oak	38.4	11.6	16.4	9.6	7.2	6.4	0.4	2.0	0.0	0.0	0.0	8.0
Maven	41.1	4.5	10.7	3.6	10.7	11.6	0.0	0.9	0.0	0.0	0.0	17.0
PDFBox	25.0	8.8	6.1	2.7	6.1	10.8	2.0	2.7	0.0	0.7	0.0	35.1
Solr	31.8	9.9	9.9	7.1	7.5	8.3	2.0	2.4	0.8	0.0	0.0	20.2
Wicket	30.8	3.0	15.0	7.5	9.0	9.8	0.0	7.5	0.0	0.0	0.0	17.3
Average	29.4	11.6	15.5	7.7	8.7	9.1	0.9	2.4	0.5	0.3	0.0	13.9

TABLE III: Proportion of associated bugs that are fixed in the same commit for each selected project.

Project	#BugPairA	#BugPairSC	%BugPairSC
Accumulo	5	0	0.0
ActiveMQ	14	4	28.6
Calcite	16	5	31.2
Hadoop	44	0	0.0
HBase	28	7	25.0
Hive	67	3	4.5
Jackrabbit Oak	20	5	25.0
Maven	19	5	26.3
PDFBox	52	7	13.5
Solr	51	35	68.6
Wicket	23	6	26.1

explicitness ($\leqslant 0.5$), and 13.9% of bug pairs are perfectly associated.

B. AE of Different Association Types (RQ2)

We calculated the average AE values of different association types for each selected project as shown in TABLE IV, where the last row is the average for all projects. Association types Blocked/Blocker, Cloners, Dependent, and Required have relatively small AE values on average, while Container has a relatively large AE on average. Association types ChildIssue/ParentFeature and Dependency each has only one bug pair, the average AE values for these two types do not make much sense.

We ran Mann-Whitney U tests to examine if there are significant differences on AE between bug pairs of different association types. Since ChildIssue/ParentFeature and Dependenccy have only one bug pair, and Duplicate has even no bug pair, we did not run the tests for these three association types. The test results are shown in TABLE V, where cells with *p-value*$<$0.05 are filled in gray. Specifically, a gray-filled cell with a number in bold (resp. regular) indicates that the AE of bug pairs with the association type of the corresponding row is significantly larger (resp. smaller) than the AE of bug pairs with the association type of the corresponding column. The main points are reported as follows: (1) The average AE of bug pairs with association type Blocked/Blocker is significantly smaller than the average AE of bug pairs with association types Container, Problem/Incident, Required, and Supercedes. (2) The average AE of bug pairs with association type Cloners is significantly smaller than the average AE of bug pairs with association types Container, Problem/Incident, Reference/Related, Regression, and Supercedes. (3) The average AE of bug pairs with association type Container is significantly larger than the average AE of bug pairs with association types Dependent, Incorporates, Reference/Related, Regression, and Required. (4) The average AE of bug pairs with association type Dependent is significantly smaller than the average AE of bug pairs with association types Problem/Incident, Reference/Related, Regression, and Supercedes.

Summary: Relatively speaking, bug pairs with association types Blocked/Blocker, Cloners, Dependent have relatively weak association explicitness, while bug pairs with association type Container have relatively strong association explicitness.

V. Discussion

A. Interpretation of Study Results

RQ1: (1) As we can see from TABLE II, the association explicitness of each project shows a distribution pattern: a significant proportion of bug pairs are not explicitly associated, a majority of bug pairs are explicitly associated to some extent, and a non-trivial proportion of bug pairs are perfectly associated in the bug-fixing commits. (2) On average, 70.6% of the bug pairs are with an association explicitness larger than 0, which indicates that the association of a majority of manually-associated bug pairs can be traced with some clues in the code change history. In contrast, on average 29.4% of bug pairs in the selected projects are not explicitly associated, indicating that the association of a minority of manually-associated bug pairs cannot be reflected in the code change history. (3) 13.9% of the bug pairs are fixed in the same commit(s), and a potential reason is that those bug pairs may be in specific types of associations so that the bug pairs tend to be fixed simultaneously in the same commit(s). For instance, in project *ActiveMQ*, bugs *AMQ-2967* and *AMQ-2959* are in the association of Supercedes; by definition, when *AMQ-2967* is fixed, *AMQ-2959* should also be fixed, and consequently these two bugs are fixed in the same commit.

RQ2: (1) The association explicitness of bug pairs with association type Container is significantly larger than that of bug pairs with most of other association types. One potential reason for this phenomenon is that the two bugs in an association of Container tend to be fixed in the same commit(s) according to the definition of Container. (2) The association

TABLE IV: Average AE values of different association types for each selected project.

Project	Blocked/ Blocker	Child/Issue/ Parent/Feature	Cloners	Container	Dependency	Dependent	Duplicate	Incorporates	Problem/ Incident	Reference/ Related	Regression	Required	Supercedes
Accumulo	0.133		0.063			0.069		0.083	0.271	0.147	0.230		0.056
ActiveMQ						0.159		0.353	0.500	0.256	0.370	0.109	0.385
Calcite	0.110		0.292	0.667		0.142		0.206	0.410	0.245	0.341	0.071	0.643
Hadoop	0.068	1.000	0.142			0.269		0.333	0.378	0.345	0.291	0.148	0.200
Hbase	0.218		0.000			0.305		0.546	0.271	0.249	0.174	0.322	0.500
Hive	0.166		0.202	1.000	0.000	0.219		0.099	0.544	0.317	0.371	0.195	0.325
Jackrabbit Oak	0.129		0.177	0.125		0.108		0.500	0.089	0.218	0.289	0.414	0.000
Maven	1.000					0.398			0.778	0.246			1.000
PDFBox	1.000		1.000	1.000		0.193			0.500	0.470	0.750		0.500
Solr	0.161		0.333	0.639		0.280		0.273	0.202	0.362	0.339		0.235
Wicket	0.167		0.357			0.429		0.000	0.281	0.341	0.375		
All	0.180	1.000	0.235	0.668	0.000	0.224		0.287	0.345	0.302	0.318	0.233	0.384

TABLE V: *P-values* of Mann-Whitney U tests between different association types.

Association type	Blocked/ Blocker	Cloners	Container	Dependent	Incorporates	Problem/ Incident	Reference/ Related	Regression	Required	Supercedes
Blocked/Blocker	-	0.755	0.001	0.465	0.151	<0.001	1.097	2.733	0.012	0.001
Cloners	0.755	-	0.003	0.419	0.210	0.021	0.034	0.007	0.098	0.019
Container	**0.001**	**0.003**	-	**0.002**	**0.010**	0.057	**0.009**	**0.017**	**0.010**	0.093
Dependent	0.465	0.419	0.002	-	0.394	0.008	0.006	<0.001	0.067	0.011
Incorporates	0.151	0.210	0.010	0.394	-	0.155	0.502	0.154	0.434	0.119
Problem Incident	**<0.001**	**0.021**	0.057	**0.008**	0.155	-	0.181	0.509	0.159	0.722
Reference/Related	1.097	**0.034**	0.009	**0.006**	0.502	0.181	-	0.072	0.942	0.130
Regression	2.733	**0.007**	0.017	**<0.001**	0.154	0.509	0.072	-	0.359	0.323
Required	**0.012**	0.098	0.010	0.067	0.434	0.159	0.942	0.359	-	0.095
Supercedes	**0.001**	**0.019**	0.093	**0.011**	0.119	0.722	0.130	0.323	0.095	-

explicitness of bug pairs with association type Dependent is significantly smaller than that of bug pairs with most of other association types. One potential reason is that the two bugs in an association of Dependent may be coupled more in code structure than in code change history.

B. Implications

Most bug pairs are explicitly associated to certain degree, which implies that the code change history is a helpful resource for developers to deal with bugs and their impact. A significant proportion of bug pairs are not explicitly associated at all in each project, which implies that it is necessary to turn to other sources (e.g., code structure dependencies) for bug analysis.

VI. Conclusions

This work investigates to what extent manually associated bugs can be explicitly traced in their code change history, and whether there is significant difference on the association explicitness between bugs in different types of association. To answer these questions, we performed an empirical study on 11 Apache OSS projects, and obtained the following findings: (1) Around 71% of manually-associated bug pairs can be traced with overlapped changed source files in the code change history, and in contrast, around 29% are not explicitly reflected in the code change history at all. (2) Bug pairs with association types Blocked/Blocker, Cloners, and Dependent have relatively weak association explicitness, while bug pairs with association type Container has relatively strong association explicitness.

References

[1] A. Nicholson and G. Jin L.C., "Issue link label recovery and prediction for open source software," in *Proceedings of REW*. IEEE, 2021, pp. 126–135.

[2] P. Heck and A. Zaidman, "Horizontal traceability for just-in-time requirements: the case for open source feature requests," *Journal of Software: Evolution and Process*, vol. 26, no. 12, pp. 1280–1296, 2014.

[3] D. Falessi, J. Roll, J. Guo, and J. Cleland-Huang, "Leveraging historical associations between requirements and source code to identify impacted classes," *IEEE Transactions on Software Engineering*, vol. 46, no. 4, pp. 420–441, 2020.

[4] P. Rempel and P. Mäder, "Preventing defects: The impact of requirements traceability completeness on software quality," *IEEE Transactions on Software Engineering*, vol. 43, no. 8, pp. 777–797, 2017.

[5] B. Kucuk and E. Tuzun, "Characterizing duplicate bugs: An empirical analysis," in *Proceedings of SANER 2021*. IEEE, 2021, pp. 661–668.

[6] M. T. Tomova, M. Rath, and P. Mäder, "Poster: Use of trace link types in issue tracking systems," in *Proceedings of ICSE Companion*. IEEE, 2018, pp. 181–182.

[7] D. Ståhl, K. Hallén, and J. Bosch, "Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework," *Empirical Software Engineering*, vol. 22, no. 3, pp. 967–995, 2016.

[8] T.-D. B. Le, M. Linares Vásquez, D. Lo, and D. Poshyvanyk, "Rclinker: Automated linking of issue reports and commits leveraging rich contextual information," in *Proceedings of ICPC*. IEEE, 2015, pp. 36–47.

[9] Z. Li, X. Qi, Q. Yu, P. Liang, R. Mo, and C. Yang, "Multi-programming-language commits in oss: An empirical study on apache projects," in *Proceedings of ICPC*. IEEE, 2021, pp. 219–229.

[10] Z. Li, Q. Yu, P. Liang, R. Mo, and C. Yang, "Interest of defect technical debt: An exploratory study on apache projects," in *Proceedings of ICSME*. IEEE, 2020, pp. 629–639.

Data Selection for Cross-Project Defect Prediction with Local and Global Features of Source Code

Xuan deng
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
xuan_deng@qq.com

Peng He*
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
penghe@hubu.edu.cn

Chun Ying Zhou
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
zcy9838@qq.com

***Abstract*[1] :** *An open challenge for cross-project defect prediction (CPDP) is how to select the most appropriate training data for target project to build quality predictor. To our knowledge, existing methods are mostly dominated by traditional hand-crafted features, which do not fully encode the global structure between codes nor the semantics of code tokens. This work is to propose an improved method which is capable of automatically learning features for representing source code, and uses these feataures for training data selection. First, we propose a framework ALGoF to automatically learn the local semantic and global structural features of code files. Then, we analyze the feasibility of the learned features for data selection. Besides, we also validate the effectiveness of ALGoF by comparing with the traditional method. The experiments have been conducted on six defect datasets available at the PROMISE repository. The results show that ALGoF method helps to guide the training data selection for CPDP, and achieves a 48.31% improvement rate of F-measure. Meanwhile, our method has statistically significant advantages over the traditional method, especially when using both the local semantic and global structural features as the representation of code files. The maximum improvement of F-measure can reach 42.6%.*

Keyword : cross-project defect prediction; semantic feature; structural feature; software quality; representation learning.

I. INTRODUCTION

The main purpose of cross-project defect prediction (CPDP) is to predict defect-prone files in a project based on the defect data collected from other projects.. Peter et al. [1] proposed that a major issue in CPDP is how to find the appropriate training data set (TDS) for the target project. That is, the selection of high-quality cross-project training data is a key breakthrough.Nowadays,there is a growing collection of defect datasets on the Internet.Thus, the construction of an appropriate TDS is a more serious challenge for CPDP.

To address this issue, researchers in this field have attempted to characterize code files by using traditional hand-crafted features (e.g., CK, Halstead, MOOD, and McCabe's CC metrics), and guide the TDS selection based on these metric values[3-4,6]. Unfortunately, these metrics only contain statistical information of programs and require human design. As we known, programs have well-defined syntax and rich semantics hidden in the Abstract Syntax Trees (ASTs), which have been successfully extracted and used for defect prediction [7]. In addition, researchers also validated that the globally structural information extracted by network representation learning can lead to more accurate defect prediction [9-11].In other word, both the local semantic and global structural information of source code files may affect the selection of TDS in CPDP.

Thus,we propose a new framework called ALGoF to Automatically learn the Local semantic (fine-grained) and Global structural (coarse-grained) Features of code files for data selection in CPDP and seek empirical evidence that they can achieve acceptable performance compared with the traditional method. Our contributions are summarized as follows:

- We leverage representation learning technique to automatically learn the local and global features of the program from source code files, and use to guide the training data selection in CPDP.
- The results on six projects show that the proposed ALGoF method can improve CPDP, compared to using traditional hand-crafted features. The combination of global and local features has greater impact on the improvement of prediction performance.

The rest of this paper is organized as follows. Section 2 is a review of related work. Sections 3 describes the proposed approach.Section 4 is the detailed experimental setups, and Section 5 shows and discusses the experimental results. Finally, Section 6 concludes the work and presents the agenda for future work.

II. RELATED WORK

A. Data Selection for CPDP

In software engineering, CPDP has drawn wide attention and many studies are carried out to explore the strategies of training data selection. Peters et al. [1] proposed a filter guided by the structure of content-rich source project data and achieved great performance.To obtain a comprehensive evaluation, Bin et al. [8] even conducted a thorough experiment to compare nine relevancy filters on 33 datasets.Hosseini et al. [6] further showed that the selection of training data can lead to

[1]DOI reference number: 10.18293/SEKE2022-086

better performance in CPDP, and concluded that search-based methods combined with feature selection was a promising way.

From the above rich results, there is a commonality that is the defect data used are represented by traditional hand-crafted software metrics.

B. Representation Learning in Software Engineering

Representation learning has been widely applied to feature learning.In software engineering, some algorithms have been adopted to the Abstract Syntax Trees (ASTs) representation of source codes. For example, Wang et al. [7] leveraged Deep Belief Network to automatically learn semantic features from token vectors extracted from programs'ASTs and further validated that the learned semantic features significantly improved defect prediction.Besides, some studies have demonstrated the effectiveness of structural features in improving defect prediction. For example, Qu et al. [10] used network embedding technique, node2vec, to automatically learn to encode dependency network structure into low-dimensional vector spaces to improve software defect prediction. Zeng et al. [11] also recently analyzed the influence of network structure features of code on defect prediction.

Existing studies have verified the usefulness of semantic and structural features on defect prediction, but do not involve the analysis of the impact on data selection in CPDP.

III. APPROACH

This section introduces the entire framework of ALGoF method in detail, mainly comprised of three parts: automatically learning the semantic and structural features of code files, training data selection and defect prediction (Figure 1). First, we extract the dependencies between the classes from the source code files to construct a class dependency network. Then, we perform network embedding learning on the CDN to generate the global structural features of classes. Meanwhile, we leverage a Convolutional Neural Network (CNN) to automatically learn semantic features using token vectors extracted from the class files' abstract syntax trees (AST). Second, combine the structural and semantic features of the class obtained in the previous step, and use them as the representation of the class file. After that, the similarity scores between source file instances and each target file instance are recorded and used to guide how to select appropriate source file instances for cross-project defect prediction. Finally, we use the resulting source file instances to train the predictor and test on the target instances.

A. Generation of local semantic features

1) Parsing AST

As previous study [9] has shown, AST can represent the semantic information of source code file with the most appropriate granularity.

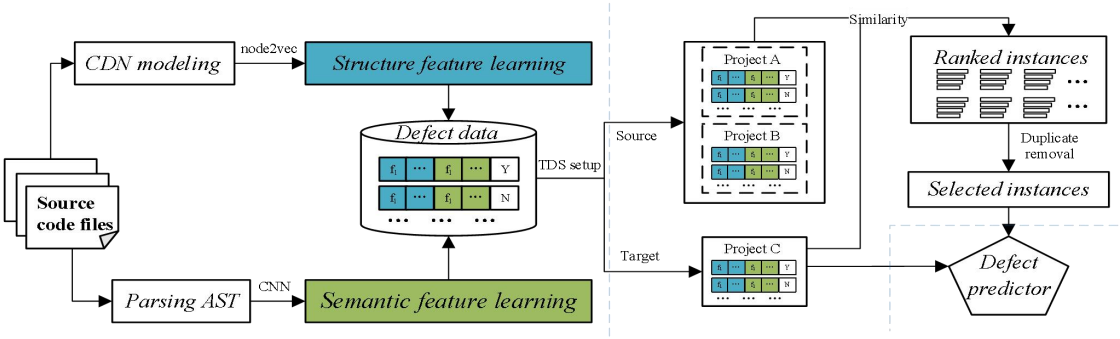


Figure 1. The entire framework of ALGoF method

We first parse the source code files into ASTs by calling an open-source python package javalang. Given a path of the source code, the token sequences of all files will be output.As treated in [7], we only select three types of nodes on ASTs as tokens: (1) nodes of method invocations and class instance creations; (2) declaration nodes, i.e., method/type/enum declarations; (3) control flow nodes, such as while, if, and throw. For more details, please refer to our previous work [11].

Then,we convert the extracted token sequences into the numerical token vectors.We append 0 to each integer vectors to make each files' lengths consistent with the longest vector.Note that, to filter out infrequent tokens, we only encode tokens occurring three or more times, the others denote as 0.

2) Building CNN

After encoding and preprocessing token vectors,we exclude the input and output layers. We train the CNN model with four layers: an embedding layer (turn integer token vectors into real-valued vectors of fixed size), a convolutional layer, a max-pooling layer, and a fully connected layer.

Given a project P, assume tha it contains n source code files, all of which have been converted to integer token vectors $x \in \mathcal{R}^l$, where l is the length of the longest token sequence.Through the embedding layer, each file becomes a real-value matrix $X_{l\times d}$. As the input of convolutional layer, a filter $\mathcal{L} \in \mathbb{R}^{h\times d}$ is applied to a region of h tokens to produce a new feature.

Then, the max-pooling operation is performed on the mapped features and the maximum value $\widehat{F} = max\{f\}$ is taken as the feature corresponding to that particular filter $\mathcal{L}$. Usually, multiple filters with different region sizes are used to get multiple features. Finally, a fully connected layer further generated the local features.

B. Generation of Global structural Features

Before applying network embedding to represent global structural features of source codes, it is necessary to build a Class Dependency Network. As did in [11], we use DependencyFinder API to parse the compiled source files (.zip or .jar extensions) and extract their relationships using a tool developed by ourselves. With the CDN, we further perform embedding learning using the *node2vec* method. For more details on *node2vec*, please refer to the literature [2].

C. Training Data Selection

The general training data selection process consists of three steps: candidate TDS setup, ranking and remove duplicate.For more details, please refer to our previous work [4].In order to characterize each code file, the two types of features obtained above are connected and marked with defect label, so as to generate the defect data set required for subsequent tasks.

An instance is characterized as a feature vector, and each target instance as an input. A similarity index (e.g., cosine similarity) is applied to construct a model for ranking source instances based on the given target instances. For instance, the cosine similarity between a source instance I_p and an input target instance I_q is computed via their vector representations:

$$Sim(I_p, I_q) = \frac{\vec{I_p} \cdot \vec{I_q}}{\|I_p\| \times \|I_q\|} = \frac{\sum_{i-1}^{n} (f_{pi} \times f_{qi})}{\sqrt{\sum_{i=1}^{n} f_{pi}^2} \times \sqrt{\sum_{i=1}^{n} f_{qi}^2}} \quad (1)$$

Where $\vec{I_p}$ and $\vec{I_q}$ are the feature vectors for instances I_p and I_q respectively. f_i represents the i^{th} feature value of the features.

For each target instance, the top-k (k=10) source instances are ranked by the Sim values and returned. Hence, the finally selected TD is composed by integrating the set of top-k source instances of each target instance (i.e., the duplicate instances are removed to maintain uniqueness).

IV. Experiment Setup

A. Dataset

In this paper, 6 defect datasets from the PROMISE repository[2] were selected for validation. Detailed information on the datasets is listed in Table 1, where *#files* and defect rate are the number of files and the percentage of defective files, respectively.

B. Experimental Design

To make a comparison between the traditional hand-crafted features and automatically learn features in our paper, four scenarios will be considered in our experiments.

(i)THC represents predictor based on the traditional hand-crafted features.

(ii)ALoF represents predictor based only on the local semantic features.

(iii)AGoF represents predictor based only on the global structural features.

(iv)ALGoF represents predictor based on both the local and global features.

C. Classifiers and Evaluation Measures

This paper utilizes logistic regression (LR), which is widely used in the defect prediction, as the classifier. We use the default parameter settings for LR specified in Weka[3] unless otherwise specified.

To evaluate the defect prediction model's performance, we use widely adopted F-measure, which is the harmonic mean of precision and recall.

TABLE I. Details of the Datasets

Project	Releases	#files	defect rate(%)
Camel	1.4	892	17.1
Lucene	2.0	186	48.9
Poi	2.5	379	65.1
Synapse	1.1	222	27.0
Xalan	2.6	875	47.0
Xerces	1.3	446	15.0

V. Experimental Results

RQ1: *Does data selection based on ALGoF and its variants of CPDP work well?*

We first take the case where the initial source TDS without any selection is considered as a baseline, labeled as iTDS. On contrary, we label the case of TDS selection using the features learned in this paper as sTDS. Then we perform cross-project predictions in both cases mentioned above.

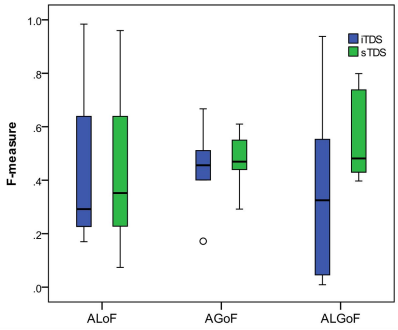


Figure 2. A comparison on F-measure of CPDP under the case of iTDS and sTDS.

TABLE II. The Improvement of F-measure of CPDP under the Case of iTDS and sTDS.

Model	iTDS	sTDS	Δ（%）
ALoF	0.292	0.352	20.55%
AGoF	0.456	0.470	3.07%
ALGoF	0.325	0.482	48.31%

Figure 2 shows that, on average across the six datasets, for CPDP with iTDS, the median F-measure are 0.292,0.456 and 0.325 respectively,while sTDS are 0.352,0.470 and 0.482 Clearly, the results verify the necessity of data selection for CPDP. The improvement rate is the most obvious in the ALGoF scenario, reaching 48.31%, followed by that of ALoF with 20.55%. The results also show that AGoF performs better than ALoF whether data selection is used or not. However, ALGoF outperforms ALoF and AGoF when only considering

[2] http://promise.site.uottawa.ca/SERepository/datasets-page.html

[3] http://www.cs.waikato.ac.nz/ml/weka/

the training data selection, indicated by the largest F-measure value in bold.

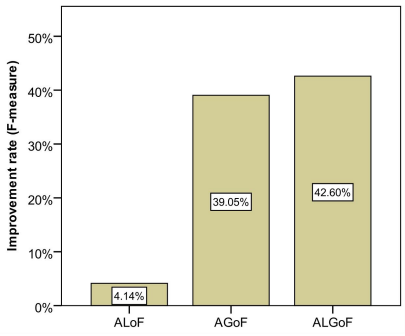


Figure 3. The improvement rate of F-measure values compared with THC scenario (with data selection).

TABLE III. THE WILCOXON SIGNED-RANK TEST AND CLIFF'S DELTA ($0.33 \leqslant |\delta| < 0.474$ MEANS THE MEDIUM EFFECTIVENESS LEVEL, AND $|\delta| \geqslant 0.474$ MEANS THE LARGE EFFECTIVENESS LEVEL [7]).

	Sig. p-value (0.05)	**cliff's delta (δ)**
ALoF_iTDS-ALoF_sTDS	0.934	-0.028
AGoF_iTDS-AGoF_sTDS	0.745	0.000
ALGoF_iTDS-ALGoF_sTDS	0.116	-0.484
ALoF_sTD-ALGoF_sTDS	0.219	-0.389
AGoF_sTD-ALGoF_sTDS	0.345	-0.083

Additionally, statistical tests assist in understanding whether a statistically significant difference between two results exists. We further utilize the Wilcoxon signed-rank test and Cliff's effect size (δ) to check whether the difference among the prediction models is significant. In Table 3, the results highlight that there are no significant differences between CPDP with iTDS and CPDP with sTDS in our experiment, indicated by all the *p*-values >0.05. However, the effectiveness level between ALGoF_iTDS and ALGoF_sTDS is large, indicated by $|\delta| = 0.484$. Besides, the effectiveness level between ALoF_sTDS and ALGoF_sTDS is medium.

TABLE IV. COMPARISON OF WILCOXON SIGNED-RANK TEST AND CLIFF'S EFFECT SIZE OF THE AUTOMATIC EXTRACTION FEATURES WITH THC.

	Sig. p-value (0.05)	**cliff's δ**
ALoF_sTDS - THC_sTDS	0.719	0.111
AGoF_sTDS - THC_sTDS	0.035	0.417
ALGoF_sTDS - THC_sTDS	0.035	0.667

In short, for the data selection task in CPDP, the features automatically learned from the source code are helpful to guide the task, so as to improve the prediction performance. In addition, the global structural feature works better than the local semantic feature, but the combination of the two is optimal. Nevertheless, data quality is more important than data quantity.

RQ2: *For CPDP data selection, which is better: automatically learned features or traditional hand-crafted features?*

In this part, we mainly compare the ALGoF method proposed in this paper with THC. As seen in Figure 3, the improvement rates of F-measure of ALoF, AGoF and ALGoF are 4.14%, 39.05% and 42.6% respectively. That is, for the data selection problem of CPDP, the use of automatically learned features to represent the source code files is better than the use of traditional hand-crafted features. In addition, Table 4 also further shows that there are significant differences between ALGoF (AGoF) and THC, indicated by the small *p*-value of 0.035 (<0.05) and the large δ value of 0.667 (>0.474). Meanwhile, according to the *p*-value of 0.719 and the δ value of 0.111, the local semantic features seem to have comparable effects to traditional source code features.

In summary, the method proposed in this paper can pick higher quality training sets for CPDP than using traditional hand-crafted features. Especially when both local semantic features and global structural features are considered.

VI. CONCLUSION

This study is to propose an improved method which is capable of automatically learning features for representing source code, and uses these feataures for training data selection. The results indicate that features automatically learned from the source code (e.g., local semantic feature and global structural feature) are helpful to guide the training data selection for CPDP.Meanwhile, compared with the case of no data selection processing, the F-measure improvement rate of ALGoF is 48.31%. In addition, the results also show that our method is significantly better than the traditional method, especially when using both the local semantic and global structural features as the representation of code files. Notedly, about 42.6% defective instances can be additionally predicted by our method.

In the future, we would like to extend our automatically feature generation approach to C/C++ projects for CPDP. In addition, it would be promising to leverage our approach to guide heterogeneous defect prediction.

REFERENCES

[1] F Peters, Menzies T , Marcus A . Better cross company defect prediction[C]// in Proceedings of the 10th MSR, 2013:409–418.

[2] A. Grover and J. Leskovec, node2vec: scalable feature learning for networks[C], in Proc. of ACM SIGKDD Inte. Conf. on Know. Dis.& Data Min., San Francisco, CA, USA, August 2016.

[3] Ryu D , Jang J I , Baik J , et al. A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction[J]. Journal of Computer Science & Technology, 2015, 30(005):969-980.

[4] He P , He Y , Yu L , et al. An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data[J]. Mathematical Problems in Engineering, 2018, (PT.6):2650415.1-2650415.18.

[5] Hosseini S., Turhan B., Mäntylä M.: A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. Inf. Softw. Technol. 2018,95, 296–312.

[6] Hosseini S , Turhan B . A comparison of similarity based instance selection methods for cross project defect prediction[C]. 36th ACM/ SIGAPP Symposium on Applied Computing, 2021:1455-1464.

[7] Wang S , Liu T , Nam J , et al. Deep Semantic Feature Learning for Software Defect Prediction[J]. IEEE Transactions on Software Engineering, 2020,46(12):1267-1293.

[8] Y. Bin, K. Zhou, H. Lu, Y. Zhou, B. Xu, Training data selection for cross-project defection prediction: Which approach is better?[C]. Int. Sym. on Emp. Soft. Eng. & Meas. ,2017:354–363.

[9] A. V. Phan, M. L. Nguyen, and L. T. Bui, Convolutional Neural Networks over Control Flow Graphs for Software Defect Prediction, 2018, https://arxiv.org/abs/1802.04986.

[10] Qu Y , Liu T , Chi J , et al. node2defect: using network embedding to improve software defect prediction[C]. The 33rd ACM/IEEE Inte. Conf. on Automated Software Engineering, 2018:844-849.

[11] Zeng C , Zhou C Yi , Lv S K , et al. GCN2defect：Graph Convolutional Networks for SmoteTomek-based Software Defect Prediction[C]. The 32nd Inter. Sym. on Software Reliability Engineering (ISSRE 2021)

RIRCNN: A Fault Diagnosis Method for Aviation Turboprop Engine

Lei Li[1], Zhe Quan[✉,1], Zixu Wang[1], Tong Xiao[1], Xiaofei Jiang[1], Xinjian Hu[1] and Peibing Du[2]

[1]College of Information Science and Engineering, Hunan University, Changsha, China
{aleilei, quanzhe, wangzixu, xiaotong18, jiangxiaofei,
huxinjian}@hnu.edu.cn
[2]Northwest Institute of Nuclear Technology, Xi'an 710024, China
dupeibing1@nint.ac.cn

Abstract

Aero-engine is the 'heart' of the aviation aircraft. Practical failure prediction of aero-engines is difficult due to the performance degradation covered by the continuous switching between various operating conditions. In order to solve the above problem, we propose a new type of aero-engine fault diagnosis model–RIRCNN (Residual Independently Reccurent and Convolutional Neural Network). It can process long sequences, and has superior feature extraction effect. We gather flight data sets through ground bench experiment of the aviation turboprop engine, and intensively conduct comparative experiments to evaluate the effectiveness of our model. The verification results demonstrate that our model can achieve excellent performance compared with other available baseline models.

Key words- Air Circuit Fault Diagnosis; Aviation Turboprop Engine; Neural Networks

1 Introduction

The safety of the aircraft is very important to ensure the military and people's livelihood. Aviation turboprop engines are mainly used in military transport aircraft. Compared with other types such as gas turbine engines, turboprop engines have a harsher working environment. The air circuit is the most prone to failure, so it is important to detect its failure. The traditional model-based, data-driven, and knowledge-based diagnostic methods are not accurate and economical. Thus, it is necessary to carry out research on key technologies such as feature data extraction, intelligent fault diagnosis and turboprop engine health state prediction. The artificial intelligence algorithm has strong reasoning ability and generalization ability, and has inherent advantages for complex engine fault diagnosis. Some intelligent algorithms have been applied in advanced Aero-engine health management systems, such as Deep Belief Network [1], LSTM (Long Short-Term Memory) [2] and Data Mining Techniques [3]. Although effective in different ways, these methods all have certain drawbacks.

Our purpose is to perform diagnostics on air path fault data from aero-turboprop engines. Considering that the essence of engine flight data is a kind of regular time series data, the model used for processing time series data in machine learning is given priority. RNN (Recurrent Neural Network) has been proposed as a solution to process time series and widely used and improved. LSTM [4] is a variant of RNN, proposed for solving the gradient disappearance and explosion problems. However, for data dealing with long time steps LSTM still has the limitation, it can only discriminate tokens in a small range and have difficulty capturing long-term dependencies. The flight conditions are constantly changing, and the flight duration is not invariable, it may cause the amount of series data be very long. So a more suitable model is needed.

In this paper, after investigating a lot of methods we propose a new model named RIRCNN. It is a new model based on the residual combination of IndRNN (Independently Recurrent Neural Network) [5] and CNN (Convolutional Neural Networks) [6]. Our main contributions are shown below:

(1) In the aviation turboprop engine bench experiment, we simulate flight states of normal and component failures under different working conditions, and obtain the data sets.

(2) We propose a novel model called RIRCNN that can process for long-series time series data and extract global feature fast.

(3) We conduct extensive experiments and verify that

DOI reference number: 10.18293/SEKE2022-112

our model outperforms other available methods in air circuit fault diagnosis of aero-turboprop engines.

2 Related Work

Since the aviation turboprop engine is mostly used in the military transportation bureau, its technology and data involve secrecy. Only few public information and research are publicly available. Almost of the existing more cutting-edge artificial intelligence method resea are based on some public civil aviation turbine engine data. For example, a study [7] utilized a DFC and LSTM to established an offline health flight state estimation model and a degradation trend prediction model. Another group studied out a transfer learning method based on CNN and SVM for gas turbine fault diagnosis [8]. Zhou [9] employed a Res-BPNN and introduced the method of maximizing the domain confusion loss based on the adversarial mechanism in the experiment, so that the features learned from different domains are as close as possible and reduce the distribution difference of each aero-engine model.

Therefore, in order to conduct research based on turboprop engine failure data, it is necessary to obtain the corresponding data of the relevant model engines first. For example, the Australian Aviation and Navigation Research Laboratory took the F404 turbofan engine as the object, injected corresponding faults into several components such as the variable geometric angle and nozzle area of the compressor. And in this way, they finally obtained the simulated fault flight data.

In this paper, we obtain the data set through the ground simulation experiment of aviation turboprop engine, which made up for the shortcoming of insufficient data of this type of engine failure. Then we design a neural network named RIRCNN which can classify the data and detect faults by extracting the time series features and global features of the data.

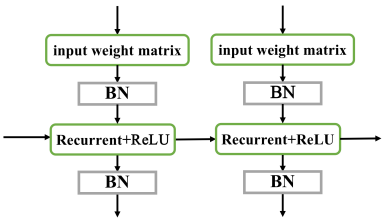
3 Approach

How IndRNN implements a neuron-independent architecture within a layer and solves the gradient problem are described in Subsection 3.1; the newly proposed model RIRCNN in the paper is introduced in Subsection 3.2.

3.1 Principle of IndRNN

Traditional RNNs models map the hidden states to outputs via the following recursive equation, it shares a weight $\mathbf{W}$ at each stage and its final output can be represented by $f[\mathbf{W}...[\mathbf{W}f[\mathbf{W}f_i]]]$. Obviously seen from the cumulative formula: when the gradient to be solved in reverse, if

Figure 1. IndRNN basic model.



the derivative of f is not 1 or 0, it is easy to cause a gradient problem. Therefore, IndRNN is trying to introduce non-saturating activation functions ReLU to stack multiple layers of IndRNN to build very deep networks. The basic IndRNN structure is shown in Figure 1. Every neuron of IndRNN only receives information from the input and its own hidden state at the previous time step, which enables each neuron in the same layer can independently process a spatial-temporal pattern. Different neurons can be cross-correlated by stacking two or more layers, in which case each neuron in the next layer processes the output of all neurons in the previous layer. The hidden state calculation formula specific to a single neuron in the hidden layer of IndRNN is as follows:

$$\mathbf{h}_t = f(\mathbf{w}_{n,\mathbf{x}_t}\mathbf{x}_t + \mathbf{w}_{n,\mathbf{h}_t} \odot \mathbf{h}_{n,t-1} + \mathbf{b}) \tag{1}$$

where $\odot$ represent the Hadamard product, $\mathbf{w}_{n,\mathbf{x}_t}$ is the n^{th} row of the input weight matrix, and $\mathbf{w}_{n,\mathbf{h}_t}$ is the recurrent weight matrix in hidden layer, respectively. For the back-propagation of the temporal gradient of each layer, since there is no interaction between neurons in the layer, the gradient of each neuron can be calculated independently. For the nth neuron $\mathbf{h}_t$, assuming the output at time step T is $\mathbf{J}_n$, the gradient back-propagated to time step t is

$$\begin{aligned}\frac{\partial \mathbf{J}_n}{\partial \mathbf{h}_{n,t}} &= \frac{\partial \mathbf{J}_n}{\partial \mathbf{h}_{n,T}} \prod_{k=t}^{T-1} \frac{\partial \mathbf{h}_{n,k+1}}{\partial \mathbf{h}_{n,k}} \\ &= \frac{\partial \mathbf{J}_n}{\partial \mathbf{h}_{n,T}} \prod_{k=t}^{T-1} f_{n,k+1}(\mathbf{w}_{n,k+1} \odot \mathbf{h}_{t-1} + \mathbf{b}_n) \\ &= \frac{\partial \mathbf{J}_n}{\partial \mathbf{h}_{n,T}} \mathbf{w}_{n,\mathbf{h}_t}^{T-t} \prod_{k=t}^{T-1} f_{n,k+1}\end{aligned} \tag{2}$$

where $f_{n,k}$is the derivative of the activation function such as ReLU and Tanh, which shows its gradient $\mathbf{w}_{n,\mathbf{h}_t}^{T-t} \prod_{k=t}^{T-1} f_{n,k+1}$ is directly depends on the value of the recursive weight matrix $\mathbf{w}_{n,h_t}$. When using ReLU as activation function(the result is the constant 0 or 1). Assuming the maximum gradient value to ensure that the gradient doesn't explode is γ. So the range of $|\mathbf{w}_{n,\mathbf{h}_t}|$ can be represented as $[0, \sqrt[T-t]{\gamma}]$. In the case of $|\mathbf{w}_{n,\mathbf{h}_t}| = 0$, the neuron only regards the information from the current input and does not

retain any past memory. This method basically maintains the gradient within an appropriate range and does not affect the gradient backtracked through the neuron. It avoids the errors caused by the commonly used gradient clipping method. By stacking the basic IndRNN structures, it is possible to build a deep network that can even handle sequences over 5000 time steps.

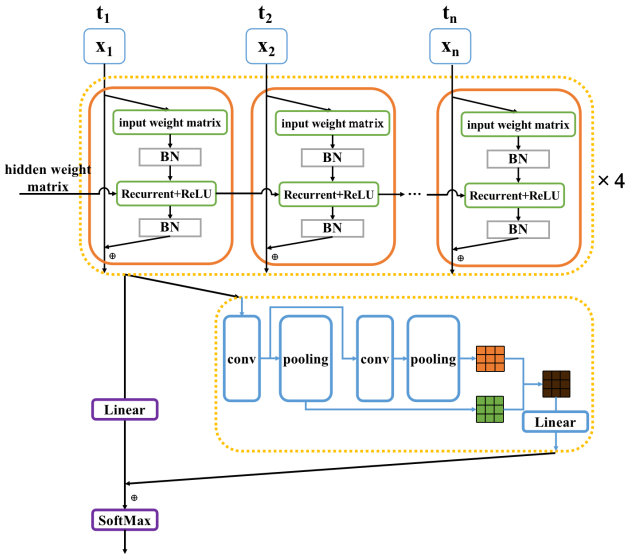
3.2 Structure of RIRCNN

As inferred by Li [5], neurons in first layer of IndRNN mainly sequence position information, one neuron in second layer aggregates input into long-term memory, while other neurons usually retain their state or process short-term memory. Therefore, in order to extract time series characteristics and state parameter characteristics of the aviation turboprop engine air circuit time series fault data as accurately as possible, we need to design a model with more than two layers. When a multi-layer IndRNN network stacked, the neurons in each layer contain the parameter characteristics of all neurons in the previous layer, and the output of the network contains the feature extraction results of all neurons in the hidden layer. We finally superimpose 4-layer IndRNN with 512 neurons in each layer. After the IndRNN model, if simply stacking the fully connected layer and the dropout layer to extract the classification results, although the probability distribution of the desired format can be obtained, the randomly discarded neuron information may cause the loss of some important information. It makes the classification precision exist a certain bottlenecks.

Therefore, in order to eliminate defects, this paper uses IndRNN as a general design for processing time series data to extract time series features. Then a convolutional neural network structure is introduced as a classifier, and the output tensor of IndRNN is used as the input of CNN. The probability distribution of the fault category can be obtained after the output of the convolution calculation. CNN adapts to data extraction features by combining convolutional layers and pooling layers. As the number of network layers increases, the corresponding extracted features are more complex, also, the receptive field is larger. When CNN calculates, the weight information of neurons in different positions is shared, and the global features of the input data can be extracted by integrating the information. These features make CNN as a classifier in the line with the purpose of digital data classification in this paper. IRCNN is a simple serial combination structure of 4-layer IndRNN and 2-layer CNN. However, simply superimposing CNN may cause overfitting of the model and reduce the diagnostic precision. Therefore, further research is needed.

Residual network proposed by He [10] is to address the degradation problem of Deep networks. The structure of residual learning is somewhat similar to a "short circuit" in

Figure 2. RIRCNN structure.



a circuit. It is to directly transfer concepts captured by previous layer to next layer. Our proposed RIRCNN residual connect 4-layer IndRNN and 2-layer CNN shown as Figure 2. We introduce residual connections between all network layers, and add the output data of the previous layers and the output data of the latter layers by weight directly. Explaining in principle, the stacked layers only do the identity mapping without increasing the parameters and computational complexity. It don't need to be re-learned every time, which improves reusability and reduces redundancy. In order to speed up the training, Batch Normalization is inserted after each layer. The output of residual connection is adjusted by fully connected linear layer, and then output to softmax activation function layer to calculate probability distribution of the fault category.

4 Experiment

4.1 Data Introduction

The existing aero-engine air circuit parameter baselines calculation model is not disclosed by the engine manufacturer as a commercial secret. And there are only a handful of fault data obtained during actual flight. In order to increase the reliability of data, we plan to obtain the data of changes in air circuit components sensed by sensors from the actual aviation flight and the ground experiments. In addition to collecting the history field data, simulating operation of the military aero-turboprop engine "WJ-XX"[1] under different

[1]The details of the aircraft cannot be disclosed due to non-disclosure agreements

working conditions in bench experiments is needed to obtain stable and long-term data.

We adjust performance parameters (such as pressure compressor Delta flow HPC/LPC-DW etc.), then we obtain the corresponding aviation turboprop engine air circuit measurement parameter changes. Aero-engines have hundreds of air circuit components, according to expert prior knowledge and historical experience, we extract the key attributes below: T1, torque, ITT, PCNF, PCNC, PCNP, P3, WFB and the working condition. We sample the data at interval (every 0.1s) according to the equipment situation and took out the data with same flight time. The data step size is maintained at about 800. Then, 8 main fault and 1 healthy states of the aviation turboprop engine air circuit are summarized: blade corrosion, blade tip wear, foreign object damage, blade fouling, insufficient opening of high-pressure/low-pressure turbine valve, improperly open/close of the turbine valve. Finally, a data set with sample size of 6149 is obtained. Training dataset and testing dataset are divided by the ratio of 8:2.

Considering the degradation trend of sensor measurement variables and some outliers are usually exist. We use the classic isolated forest algorithm to clean the data has achieved good results. Moreover, in order to make the distribution of the data more concentrated and accelerate the convergence of the model, we normalize the data. The cleaned effective data can also reduce unnecessary abnormal parameter troubleshooting.

4.2 RIRCNN Fault Classification Experiment

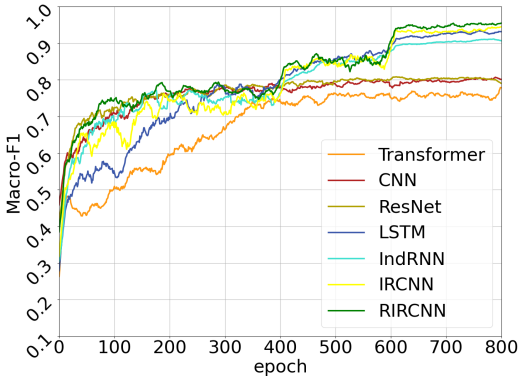
Affected by weight initialization, neural network outputs have correlated randomness. To counteract the effects of randomness, we repeat each experiment 10 times and take average of the fault classifications for comparison. We use a variety of models to compare the performance of the RIRCNN model on 6419 flight data including 9 states. Each model uses random 5.1k pieces data to train and another 1.4k pieces data to test.

The performance comparison experiments with RIRCNN include Transformer, CNN, ResNet, LSTM, IndRNN and IRCNN. During the designing of each neural network fault diagnosis model, we first use fully connected neural network to non-linearly map features extracted by each model. Then, we use function to normalize the output value of fully connected neural network to convert the probability of different categories predicted by the model. On this basis, in the training process of classifier model, the Categorical Cross Entropy Loss is used as evaluation function, Adam is used as optimization algorithm and Dropout technique is adopt to prevent the overfitting of model. In view of different structures of each model and different types of applicable data, we select the optimal configuration for each

Table 1. Memory-Usage(MemoUsg) and performance.

model	MemoUsg	Precision	Macro-F1
Transformer	2011Mib	76.82%	0.7710
CNN	1693Mib	82.38%	0.8258
ResNet	1689Mib	83.03%	0.8332
LSTM	2249Mib	92.50%	0.9277
IndRNN	1745Mib	92.53%	0.9291
IRCNN	1945Mib	94.44%	0.9466
RIRCNN	**1991Mib**	**95.73%**	**0.9610**

Figure 3. Macro-F1 of all models



in terms of number of layers and hidden units, so these parameters are not included in the assessment. We evaluate the model by calculating the failure classification precision and the multi-classification problem scoring metric Macro-F1. Table 1 records the memory required at runtime and the Precision and Macro-F1 score of fault diagnosis. Figure 3 visualizes Macro-F1 of each model.

It can be seen that the LSTM model has high precision in the task of air circuit fault data diagnosis of aero-turboprop engines, but with the highest memory consumption during model training. Compared to LSTM, RIRCNN has better performance with less memory consumption. The optimal average classification precision of LSTM and IndRNN can only reach 92.50% and 92.53%, while RIRCNN can exceed 95%, and also RIRCNN is better than other models in the Macro-F1 score. It proved that our proposed model is effective.

5 Conclusion

In this paper, we propose a novel model RIRCNN which can extract spatial information independently and extract global features and fast convergence. RIRCNN solves the limitation of RNN and its variants in terms of network depth, it can process long sequences without a large in-

crease in memory consumption. Multiple comparison experiments were conducted with existing baseline models. The result verifies that the proposed RIRCNN model is superior to the existing neural network models in the problem of air-circuit fault data diagnosis of areo-turboprop engines.

References

[1] P. Tamilselvan, Y. Wang, and P. Wang. Deep belief network based state classification for structural health diagnosis. In *Aerospace Conference*, 2012.

[2] AE Elsaid, B. Wild, J. Higgins, and T. Desell. Using lstm recurrent neural networks to predict excess vibration events in aircraft engines. In *2016 IEEE 12th International Conference on e-Science (e-Science)*, 2016.

[3] H. Gharoun, A. Keramati, M. M. Nasiri, and A. Azadeh. An integrated approach for aircraft turbofan engine fault detection based on data mining techniques. *Expert Systems*, 36(2), 2019.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[5] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. *IEEE*, 2018.

[6] Yann LeCun et al. Handwritten digit recognition with a back-propagation network. In *NIPS Conference, Denver, Colorado, USA, November 27-30, 1989*, pages 396–404.

[7] B Cwa, B Zz, C Nla, C Yca, and C Bja. A data-driven degradation prognostic strategy for aero-engine under various operational conditions. *Neurocomputing*, 2021.

[8] Shi sheng Zhong, Song Fu, and Lin Lin. A novel gas turbine fault diagnosis method based on transfer learning with cnn. *Measurement*, 137:435–453, 2019.

[9] Xingjie Zhou et al. Regression model for civil aero-engine gas path parameter deviation based on deep domain-adaptation with res-bp neural network. *Chinese Journal of Aeronautics*, 34:79–90, 2021.

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on CVPR*, pages 770–778, 2016.

Automated Unit Testing of Hydrologic Modeling Software with CI/CD and Jenkins

Levi T. Connelly, Melody L. Hammel, Benjamin T. Eger, Lan Lin
Department of Computer Science, Ball State University, Muncie, IN 47306, USA
{ltconnelly, mlhammel, bteger, llin4}@bsu.edu

Abstract

Composed of developers with diverse backgrounds in multiple disciplines, the NSF CyberWater *project team needed to research and implement effective software testing methods to improve the team's workflow efficiency and software quality. In this paper we present a practical and effective strategy for automated black-box testing of* CyberWater *modules using a Continuous Integration / Continuous Deployment (CI/CD) pipeline and the* Jenkins *automation server, Python* unittest *and* ptest*, and a novel technique we call* object-method replacement*, which isolates the backend from the front-end logic. Our experience can be adapted and extended to other research projects to mitigate the risk of programming errors and mistakes incurred through continuous development on a code repository.*

1 Introduction and Related Work

An automated software testing workflow plays a crucial role in preventing issues from creeping into the software, but the implementation of these workflows varies from project to project due to the complicated nature of software and testing. The NSF *CyberWater* project team faced some unique challenges with automated testing. In particular, their software was derived from legacy systems that integrated with *VisTrails* [7], a third-party software used to build scientific workflows as well as to support data analysis and visualization needed for hydrologic modeling and simulation. The design of *CyberWater* modules (as extended *VisTrails* modules) tightly couples the frontend and backend logic. We came up with a new technique called *object-method replacement* that allows us to address the testing problem created by the tightly coupled frontend and backend code without the need for proprietary tools for Windows-based GUI testing. Using this technique, we developed black-box unit tests using both Python *unittest* [6] and *ptest* [5] frameworks for two example modules: the `PythonCalc` module that comes with the *VisTrails* package and *CyberWater*'s `MainGenerator` module. Additionally, we researched and developed a CI/CD pipeline with a minimal set of tools supported by *Jenkins* [4] so that software developers from the domain would be able to use the pipeline on daily basis after a few training sessions efficiently and effectively.

Light-weight, agile methods and processes have drastically impacted the practice of software testing and quality control, putting testing first and in parallel with development to decrease development cost while increasing product quality [8]. State-of-the-art testing practices such as unit testing, continuous integration (CI), test-driven development, a test pyramid, test coverage analysis, etc. are considered a mandatory and indispensable part of modern software development [11]. A crucial step to leverage the benefits brought by the best industry practices, shared with successful practitioners, relies on a CI implementation [12]. Although the ideas are appealing to embrace, the choices for implementing unit testing, CI and test automation are usually heavily influenced by the particular software task and development environment, and unavoidably intimidating to software developers whose main expertise is in the domain sciences [9, 10]. The risk of software faults, programming errors and mistakes can be mitigated by continuous integration, regression testing and test automation, implemented by a testing workflow proposed here.

2 The NSF *CyberWater* Project and Challenges of Automated Unit/Module Testing

Funded by NSF, the *CyberWater* project aspires to build a new cyber infrastructure with an open data, open modeling framework and software to reduce the user time and effort required for hydrologic modeling studies, allowing related discoveries to be made sooner [2].

One of the key challenges of testing *CyberWater* modules is that the frontend and the backend are tightly coupled. Because the backend code depends on the frontend code for

DOI reference number: 10.18293/SEKE2022-074

input, the backend cannot be tested apart from the frontend without modifying the code for the backend.

In the *VisTrails* GUI (see Figure 1), each module is displayed with a set of input and output ports. Users can specify values for certain input ports for modules, while other input ports must receive data from the output of another module. Users can specify the flow of data from one module to the next by dragging a connector from the first module's output port to the next module's input port. A collection of connected modules that produces an output is called a *workflow*. When executed, the *workflow* usually generates a graph or model that visualizes the input datasets.

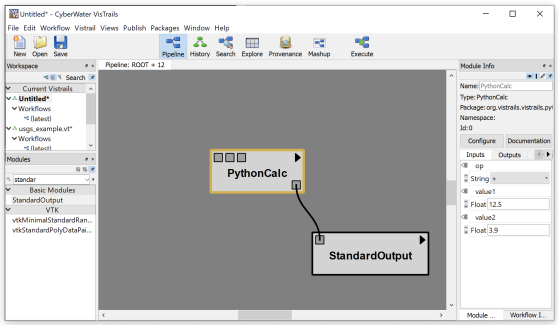


Figure 1. The *VisTrails* GUI

To retrieve the input data needed for the module's computation, the backend code of each module calls a method called `get_input`, which retrieves the input data from the module's input ports from the frontend. This dependency makes it so the backend code for *VisTrails* modules cannot be run or tested without input from the GUI without modifying the code.

3 Our Solution to Automated Unit Testing with CI/CD

3.1 A Novel Technique for Model-View Separation

We developed a technique called *object-method replacement* to achieve model-view separation. To isolate and test the backend logic we exchange a method from a specific instance of a class with a newly defined function. We define the replacement function to be nearly identical to the instance's original method, but we remove any dependencies to the frontend logic. With the backend logic isolated, we unit test the *VisTrails* module without having to modify the source code. By replacing methods in an instance of a class rather than modifying the class itself, changes to methods only affect the instance that is being altered, not the class in its entirety. Therefore, other instances of the class will not be affected by the changes.

Object-method replacement leverages the `__dict__` attribute of Python objects. This attribute is a dictionary that maps the names of each of the object's local attributes (as a string) to its corresponding value. The `__dict__` attribute is mutable, so developers can alter the values of the attributes stored in the `__dict__`, as shown below:

```
<instance>.__dict__['<attribute name>'] = <new value>
```

Because *VisTrails* modules correspond to Python classes, we can instantiate a class corresponding to a *VisTrails* module and leverage the instance's `__dict__` attribute to replace the instance's methods. This technique allows us to replace methods such as `get_input` that introduce dependencies to the GUI. We can replace the `get_input` method with a new version that simulates the GUI input, as shown below:

```
<instance>.__dict__['get_input'] = <replacement function>
```

Since we only modify the `__dict__` of a single instance of a class, this modification only changes the method for the specific instance of the class we are modifying.

Object-method replacement can be generalized with the function shown below. We call this function `mutate_method`.

```
import types

def mutate_method(obj, method_name, new_method):
    obj.__dict__[method_name] = types.MethodType(new_method, obj)
```

The function has three parameters: the target object, the name (string) of the method in that object we want to replace, and the new function. This replacement function can be structured like a normal function but with self parameters and calls to self inside the function if needed. The function we used to replace the `get_input` method is shown below.

```
from vistrails.core.modules.basic_modules import ModuleError

def get_input(self, port_name, allow_default=True):
  if port_name in self.inputPorts:
    if allow_default:
      defaultValue = self.inputPorts[port_name][0].get_output("value")
      if defaultValue is not None:
        return defaultValue
    raise ModuleError(self, "Missing value from port %s" % port_name)
```

Though this revised version of the `get_input` function is not defined in a class, the first parameter is `self`. If Python attempted to execute this revised `get_input` method without first binding it to a class, the program would throw an error. However, after using the `mutate_method` to inject the replacement function into an instance of the class we are testing, the replacement method functions the same as any other method of the class.

Object-method replacement allows developers to test the functionality of methods with dependencies to the GUI

without using specialized software to automate GUI interactions. Simply write a replacement version of the method that performs the same functionality without interacting with the GUI, instantiate a new instance of the class one is testing, and inject the new method in place of the old one.

3.2 Python Unit Testing Frameworks

There are two frameworks we have been using in tandem throughout this project: *ptest* by Karl Gong [5], and Python's built-in *unittest* library [6]. *Ptest* provides multiple advantages: testing using decorators allows for better in-code documentation with tags for grouping and a description given to every test, as well as the assertion of exceptions thrown within the decorator itself using the `expected_exceptions` parameter. These decorators include `BeforeMethod` and `AfterMethod`, which define code to be run before and after every individual test is run. They also include `BeforeClass` and `AfterClass`, which define methods to run before all of the tests run and to run after every test has finished, respectively. The `TestClass` decorator defines a class similar to creating a superclass of `unittest.TestCase` in Python's builtin, but has a parameter that allows for running tests in parallel with multiple threads. The most notable feature, however, is the test report that is generated. *Ptest*, upon finishing its tests, generates a graphic test report using HTML, CSS, and JavaScript as a visual representation of the tests that ran. It includes information about which tests failed, as well as reports of what was happening using the library's `preporter` module. It displays stack traces and descriptions of the tests that failed that were specified by the tester. This is shown in Figure 2. Python's `unittest`, alternatively, also offers distinct advantages. One problem we found with *Ptest* is that *GitHub* [3] *Actions*, *Bitbucket* [1] *Flows*, and *Jenkins* [4] *Pipelines* would all still recognize a build as passing even when *ptest*-based tests would fail. Using Python's `unittest`, however, these systems would recognize builds to be failing if the tests failed. It also provides setup and teardown methods, akin to *ptest*, and terminal output. *Ptest* also has a terminal output, shown in Figure 3.

3.3 *Jenkins* for CI/CD

One challenge that our team faced was developing a CI/CD pipeline to automatically test and merge changes made to *CyberWater* modules. We found that an efficient solution was to setup a Jenkins [4] server and link it to our *Bitbucket* repository. With *Jenkins*, developers can define jobs, which automatically run a set of tests to determine whether to automatically merge the changes from the *dev* branch into the *master* branch.

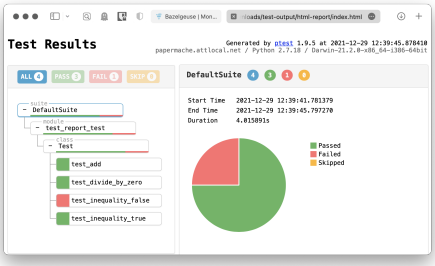


Figure 2. *Ptest*'s generated test report

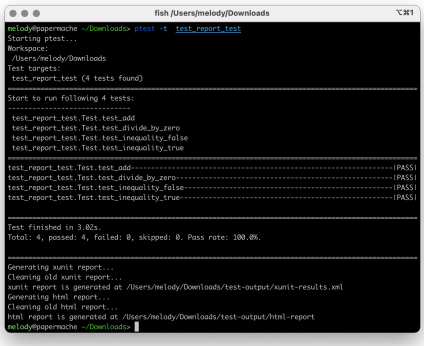


Figure 3. *Ptest*'s terminal output

After installing *Jenkins* on our Linux server, our team defined a **Freestyle project** from the dashboard. We then linked the **Freestyle project** to our *Bitbucket* repository in the **Source Code Management** section. This section allows developers to insert the URL and credentials for *Jenkins* to access the *Bitbucket* repository.

We then configured *Jenkins* to build the *dev* branch. Under **Build Triggers**, we selected **Build when a change is pushed to *Bitbucket***. This trigger configures *Jenkins* to automatically run the *job* every time someone pushes changes to the repository. Finally, we added a build step to execute a Unix shell command to run our *ptest* and *unittest* scripts.

Developers also have the option to add **Post-build Actions** which offer helpful features such as sending an email with the build results or publishing an HTML report of the build.

4 *CyberWater* Unit Testing Case Studies

4.1 `PythonCalc` Module Testing

The `PythonCalc` module is a simple module designed to incite familiarity with how to create *VisTrails* modules and the functionality of the *VisTrails* interface. It functions as a simple calculator with four possible operations:

addition, subtraction, multiplication, and division. Its input ports consist of two `Floats` or `Integers` and a `String`. All of these are wrapped as *VisTrails* modules, not Python literals. The `Float` / `Integer` input ports consist of two numbers to be operated on. The `String` input port is where the module expects an operator – this can be either '+', '-', '*', or '/'. Anything else entered will raise an exception. The module then reads the values of the number input ports, reads the `String` input port, and performs the operation on the numbers. For example, if the user entered 3, 5, and '+', the module would add 3 and 5, and return 8. The output port of the module is the result of the operation. It is typically sent to the `StandardOutput` module to print it to the persistent console that accompanies *VisTrails*. Shown in Figure 4 is an example of a possible workflow with the module, with the output in the console.

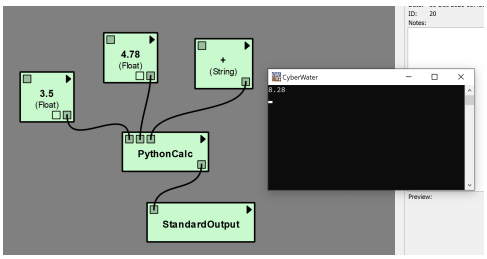


Figure 4. Usage of the `PythonCalc` module within *VisTrails*

Beginning the tests is where object-method replacement must be used. After injecting our own `get_input` method into the instance of the class we created as shown in Figure 5, we first test to ensure that all of the operators work properly, using two arbitrary numbers and an operator, asserting that the result is correct (four tests). We then test using an invalid operator to assert that an exception is raised, and we attempt to divide by zero to assert that a `ZeroDivisionException` is raised (two tests; see Figure 6). All the unit tests (using either Python *unittest* or *ptest*) pass when configured to run within *Jenkins*.

4.2 `MainGenerator` Module Testing

The `MainGenerator` module is designed as a component of the *CyberWater* framework to set up the directory for running a user's model where the simulation will take place and data will be stored. Shown in Figure 7, its input ports consist of `01_Path` and `02_GPF`, of types `Directory` and `File`, both wrapped as *VisTrails* modules. It then has 15 more input ports, named `Dataset_01`, `Dataset_02`, etc. `01_Path` is the aforementioned directory of the simulation, and `02_GPF` is a "global parameters file," designed to hold specific information needed in the simulation. The

```
@BeforeMethod(enabled=True, description="init pythoncalc & mutate_method")
def before(self):
    preporter.info("initializing pythoncalc")
    self.python_calc = PythonCalc()

    preporter.info("mutate get_input method")
    mutate_method(self.python_calc, "get_input", get_input)

    preporter.info("init constants and assign input ports")
    self.f1, self.f2, self.s1 = Float(), Float(), String()
    self.f1.setValue(3.0)
    self.f2.setValue(2.0)

def assign_parameters(self):
    self.python_calc.set_input_port("value1", self.f1)
    self.python_calc.set_input_port("value2", self.f2)
    self.python_calc.set_input_port("op", self.s1)
```

Figure 5. Setup methods for the `PythonCalc` tests using *ptest*, including object-method replacement of `get_input`

```
@Test(tags=["arithmetic", "error"],expected_exceptions=Exception)
def test_invalid_operator(self):
    self.s1.setValue('m')
    self.assign_parameters()
    self.python_calc.compute()

@Test(tags=["arithmetic", "error"],expected_exceptions=ZeroDivisionError)
def test_division_by_zero(self):
    self.s1.setValue('/')
    self.f2.setValue(0.0)
    self.assign_parameters()
    self.python_calc.compute()
```

Figure 6. Testing for an invalid operator and the divide-by-zero fault using *ptest*

`Dataset_NNs` are of type (*VisTrails*) `String`, indicating all the received datasets (with a maximum of 15) to be imported for running the user's model.

Based on the module specification provided by the development team, it begins by checking whether the directory the user has entered in `01_Path` exists. If it does, it deletes the folder and re-creates it (to ensure there are no files inside it already). If it doesn't, it creates the folder. It then copies the file specified in `02_GPF` into the new directory that was created. Then, it outputs the directory from `01_Path` in its first output port, and all the `String` objects of the `Dataset_NN` input ports are compiled into a Python `Dict<str, str>`, with the keys being the input ports, `Dataset_NNs` and the values being the `Strings` that were given to those input ports, but converted back into Python `strs`.

It is necessary to use object-method replacement to replace the `get_input` method of this module, just as with all other modules with input ports (see Figure 8). Figure 9 shows a diagram of our designed unit tests based on the `MainGenerator` module specification.

We begin by running a simple test of the `compute` method with no inputs to the module, asserting that it raises an exception. Then we proceed with three tests for the first input port `01_Path`, testing the module's behavior on a di-

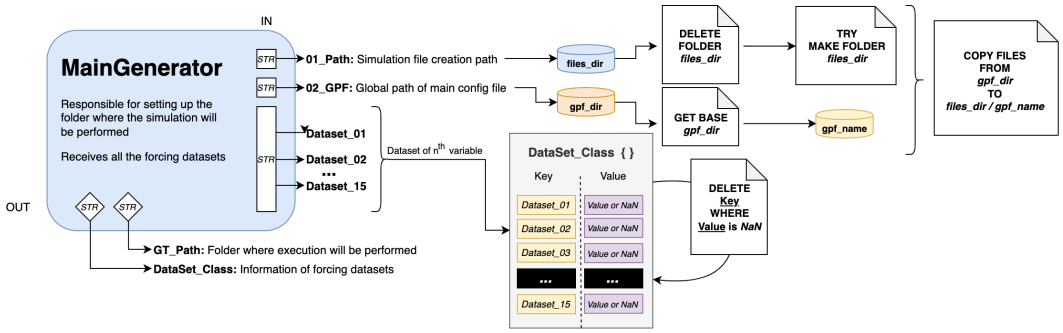


Figure 7. A diagram of the inputs and outputs of the `MainGenerator` Module

```
@BeforeMethod(enabled=True)
def __init__(self):
    preporter.info("initializing maingenerator module")
    self.maingen = MainGenerator.MainGenerator()
    mutate_method(self.maingen, "get_input", get_input)
```

Figure 8. Object-method replacement of the `get_input` method for `MainGenerator` testing using *ptest*

rectory that already exists, a directory that does not exist, and a directory that exists but for which the module doesn't have permission to access. Similarly we design three tests for the second input port `02_GPF`, assuming the file exists, or it doesn't already exist, or it exists but with no read permission. In each case, we make an assertion or assert a raised exception.

The module has two output ports `GT_Path` and `DataSet_Class` as shown in Figure 7. We test that `GT_Path` outputs the working directory we gave the module (with one test). For `DataSet_Class` we design four tests. We first test that specifically the first data set is present in the output when run, due to the special way it is handled (its input port is always shown on the GUI). We then test, giving the module two random datasets (say, chosen DataSet_01 and DataSet_04) that every dataset that was not given any input does not appear in the output. Next we give the module any dataset that isn't the first one, asserting that it exists in the output and the first one doesn't. Finally, we test that, when given no datasets, the output simply consists of an empty Python dictionary.

In all, there are twelve tests we designed, and all pass using both *ptest* and Python *unittest*. The unit tests ran automatically using *Jenkins* jobs. Figure 10 shows the HTML test report of *ptest*. Figure 11 and Figure 12 show our unit tests that test the second input port `02_GPF` using *ptest* and *unittest*, respectively.

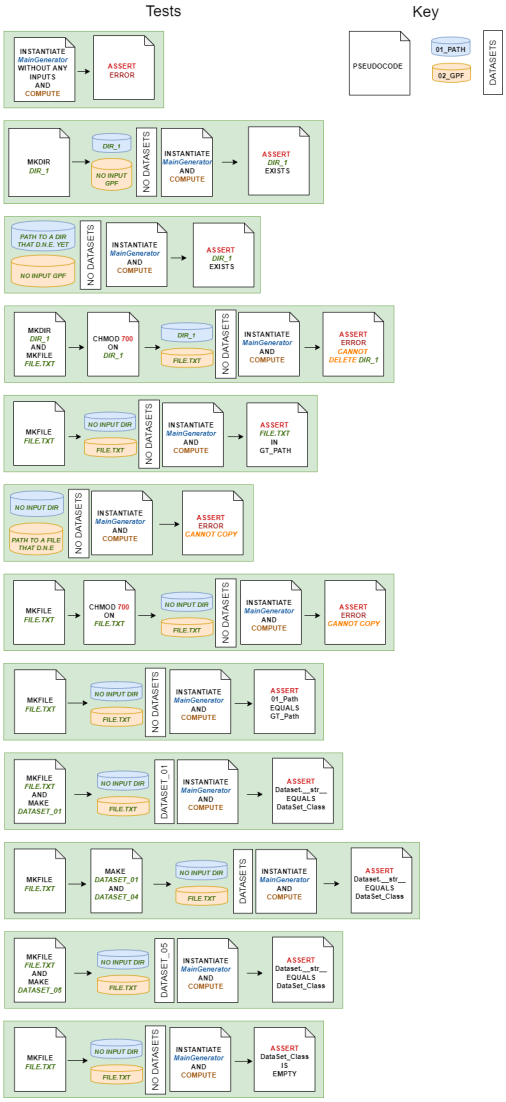


Figure 9. Design for the `MainGenerator` unit tests

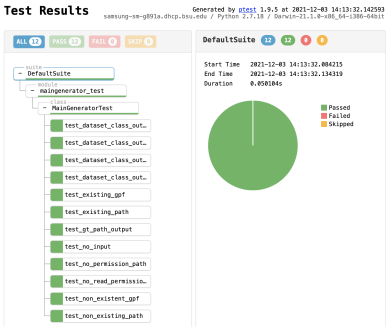


Figure 10. The *ptest* HTML output of the twelve `MainGenerator` tests

```
@Test(tags=['02_GPF'],
description='testing valid,existing gpf')
def test_existing_gpf(self):
    set_up(self.maingen, gpf=True)
    self.maingen.compute()
    assert_true(exists("test_directory/test_gpf"))

    rmtree("test_directory")
    remove("test_gpf")

@Test(tags=['02_GPF', 'error'],
description='testing non-existent file',
 expected_exceptions=IOError)
def test_non_existent_gpf(self):
    set_up(self.maingen)
    gpf = File() ; gpf.setValue("non_existent_gpf")
    self.maingen.set_input_port("02_GPF", gpf)
    self.maingen.compute()
    rmtree("test_directory")

@Test(tags=["02_GPF", "error"],
description="testing gpf with no read permission",
 expected_exceptions=Exception)
def test_no_read_permission_gpf(self):
    set_up(self.maingen)
    gpf = File()
    open("no_perms", "w").close()
    set_mode("no_perms", 700)
    gpf.setValue("no_perms")
    self.maingen.set_input_port("02_GPF", gpf)
    self.maingen.compute()
    rmdir("test_directory") ; remove("no_perms")
```

Figure 11. *Ptest* tests for `02_GPF` for `MainGenerator`

5 Conclusion and Future Work

This paper reflects on and reports our experience in applying black-box unit testing and test automation, in a CI/CD pipeline supported by *Jenkins*, to the *CyberWater* software developed for hydrologic modeling studies. We propose a novel technique called object-method replacement that provides a solution to the problem of testing software with tightly-coupled frontend and backend without the need to revise the underlying code. This is not a replacement for model-view separation, but rather a solution for software testers who want to automate testing of a legacy program with model-view separation violations. It could also be applied in scenarios where a method of an object needs to have its functionality temporarily altered. We demonstrate a testing workflow using the *Jenkins* automation server for CI/CD, and Python *unittest* and *ptest* frameworks for test automation. Future work along the line includes automated testing of more complicated *CyberWater* modules and integrated workflows. The preliminary results are promising.

```
def test_existing_gpf(self):
    set_up(self.maingen, gpf=True)
    self.maingen.compute()
    self.assertTrue( exists("test_directory/test_gpf") )
    rmtree(td);remove(tg)

def test_non_existent_gpf(self):
    set_up(self.maingen)
    gpf = File() ; gpf.setValue("non_existent_gpf")
    self.maingen.set_input_port("02_GPF",gpf)
    with self.assertRaises(IOError):
        self.maingen.compute()
    rmtree(td)

def test_no_read_permission_gpf(self):
    set_up(self.maingen, gpf=True)
    set_mode(tg, 700)
    with self.assertRaises(IOError):
        self.maingen.compute()
    remove(tg)
```

Figure 12. Python *unittest* tests for `02_GPF` for `MainGenerator`

Acknowledgments

This work was generously funded by the National Science Foundation (NSF) under Grant 1835602. It was also supported in part by an Undergraduate Honors Fellowship, funded by the Honors College, Ball State University.

References

[1] ATLASSIAN Bitbucket. `https://bitbucket.org/product/`.
[2] CyberWater. `https://www.cuahsi.org/projects/cyberwater/`.
[3] GitHub. `https://github.com`.
[4] Jenkins - Build great things at any scale. `https://www.jenkins.io`.
[5] Ptest 2.0.3 - Light test framework for Python. `https://pypi.org/project/ptest/`.
[6] Unittest - Unit testing framework. `https://docs.python.org/3/library/unittest.html`.
[7] VisTrails. `https://vistrails.org`.
[8] P. Ammann and J. Offutt. *Introduction to Software Testing, 2nd Edition*. Cambridge University Press, 2016.
[9] L. D. Couto, P. W. V. Tran-Jørgensen, R. Nilsson, and P. G. Larsen. Enabling continuous integration in a formal methods setting. *International Journal on Software Tools for Technology Transfer*, 22(6):667–683, 2020.
[10] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig. Usage, costs, and benefits of continuous integration in open-source projects. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 426–437, 2016.
[11] D. Spinellis. State-of-the-art software testing. *IEEE Software*, 34(5):4–6, 2017.
[12] S. Stolberg. Enabling agile testing through continuous integration. In *2009 Agile Conference*, pages 369–374, 2009.

Ensemble Approaches for Test Case Prioritization in UI Testing

Tri Cao[1,2,3], Tuan Ngoc Vu[2,3], Huyen Thao Le[2,3], Vu Nguyen[1,2,3,*]
[1]*Katalon LLC*
[2]*Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam*
[3]*Vietnam National University, Ho Chi Minh City, Vietnam*
{*cttri19, vntuan19, lthuyen19*}*@apcs.fitus.edu.vn, nvu@fit.hcmus.edu.vn*

***Abstract*—Test case prioritization, which focuses on ranking test cases, is an important activity in software engineering given a large number of test cases to be executed within a short period of time. Recent approaches use test execution history and test coverage as the key information for ranking prediction while reinforcement learning has the potential for improving the accuracy of prioritization. Still, each approach has its own advantages and limitations. This paper proposes a ensemble method to take advantages of several existing models by combining different them into a single one. We evaluate our ensemble models on the data sets, including sixteen projects. The results show that one of our proposed models outperforms all single models on 12 over 16 data sets.**

***Index Terms*—test case prioritization, UI testing, ensemble method**

I. Introduction

Test case prioritization (TCP) is an important activity to reduce testing effort in software projects given the lack of time for testing and delivering software releases. For user interface (UI) testing, TCP is even more important as UI testing requires more time to execute than do other types of testing [1]. TCP helps determine a subset of tests to run instead of running all available tests while aiming to satisfy a certain objective. One common objective is to maximize the chance of detecting faults.

A number of approaches have been introduced for TCP using this objective [2]–[5]. Among them, history-based and coverage-based approaches have been shown to be effective in prioritizing tests [2], [4], [5]. The history-based method uses the test verdicts from the execution history of test cases while the coverage-based focuses on optimizing the coverage of test cases using features such as length of test cases, functions, and code changes [2], [3], [6]. Bryce et al. [6] calculated the coverage of test cases based on defined parameter-value pairs. Nguyen and Le [4] proposed an approach combining test execution history and test coverage together with reinforcement learning to prioritize UI test cases. Each approach has its own strengths and limitations. For example, history-based and coverage-based approaches do not take into account the addition of test cases in regression testing since the added test cases have less information of execution history and coverage. Whereas the reinforcement learning method shows its advantage in such scenarios as it facilitates the interaction between the agent and the environment (i.e., the test cases added).

In this paper, we present an ensemble method to build test prioritization models by combining different approaches with the hope of taking advantages of these approaches. We introduce three ensemble models, including History-based ensemble by combining two single models that use execution history as the primary information, Coverage-based ensemble by using two single models that rely on coverage information, and History Coverage-based ensemble by combining History-based and Coverage-based models. The ensemble method proposed falls into the parallel category, which means each model is run independently, and the final ranking is the average of results received from those models. Furthermore, as our method focuses on UI testing, it can take advantage of the information of test steps in each test case. By that, the method uses the test step verdict to calculate the weight of each test case. To the best of our knowledge, though ensemble methods are fairly common in machine learning, their application in UI test case prioritization has not been explored before.

We perform experiments evaluating the proposed and other state-of-the-art individual approaches using 16 data sets. The results obtained from our experiments demonstrate that our best ensemble model outperforms base models on 12 over 16 data sets. This finding suggests that the ensemble method can take advantage of each individual method to produce better test prioritization performance.

The rest of our paper is organized as follows. In section II, we describe related studies on test case prioritization in the context of UI testing. Section III presents in detail our approach. Section IV describes the experimental setup, including research questions, evaluation metrics, and data sets. The experimental results are shown in section V and discussed in section VI. Section VII is our conclusion and future work discussion.

II. Related Work

Research on test suite optimization can be grouped into three main problems: test case selection, test set minimization, and test case prioritization [7]. Since our work focuses on test

* Corresponding: Vu Nguyen (nvu@fit.hcmus.edu.vn)
DOI reference number: 10.18293/SEKE2022-148

case prioritization for UI testing, this section presents related works for this specific task.

Coverage approaches [2], [6] try to order test cases to cover the target items (which can be branches, functions, objects, code changes, etc.). These approaches are straightforward and simple. However, code coverage tools which are used to track code items during execution are not always available.

History-based approaches [8], [9] use the verdicts from the execution history of test cases in different ways to prioritize test cases. Hemmati et al. grouped test cases with similar fault percentages and used diversity or random algorithms to prioritize test cases in each group [5]. Other studies proposed functions that calculate a historical value from the cost and fault severity of each test case, then order these test cases using their values [10], [11]. Noor and Hemmati calculate the similarity between the code of each test case using Hamming distance, edit distance, basic counting and combine the similarity with historical verdicts to prioritize the test cases. Wu et al. [12] consider the time window before the execution of each test case, calculate the percentage of failure for each time window, then sort the test cases based on their likelihood of failure.

Alongside new methods to represent and extract data, machine learning algorithms have been used in TCP recently because they can learn the rules automatically and therefore becomes more compatible for each project compared to traditional methods. Learning to rank algorithms, which were originally used to rank searching results or prioritize content on websites, was applied to TCP in [13]. In [4], researchers proposed a coverage graph that can utilize both the historical and coverage information of the test case. The graph can update itself after each cycle so that cycles would affect the order of test cases. Sharma and Agrawal built an information graph from UML and user story, then fed that graph into meta-heuristic algorithms [14]. Kaur et al. extracted the elements of UI in each test step, then used this data as input for traditional machine learning algorithms (SVM, decision trees, naive Bayes, etc.) [15]. For the first time, ensemble methods are applied with traditional machine learning for a general case of TCP in [16].

III. Our Approach

In this section, we describe the proposed ensemble method for UI test case prioritization.

A. Ensemble method for UI test case prioritization

Ensemble is a common method in machine learning where models need to make predictions. Combining the predictions of two or more models often gives better results than the performance of a single model [17]. This is because by joining multiple models together, weaknesses of one model are expected to be improved by other models and vice versa. In UI test case prioritization, there are different approaches with their own strengths and limitations which are able to complement each other. Hence, we apply this idea of the ensemble method for prioritizing UI test cases.

Ensemble methods in machine learning have two main paradigms: sequential ensemble methods and parallel ensemble methods [18]. In sequential ensemble methods, the dependence between base models is exploited by successively applying those base models one after another. On the contrary, parallel ensemble methods focus on the independence between base models by running them in parallel and combining the results later on. In this paper, we apply parallel methods since we do not observe any considerable dependencies between the chosen UI test case prioritization approaches.

Fig. 1 provides an overview of our model. Our ensemble method is divided into two phases: the first is to obtain predictive ranking from different models, and the second phase is to combine results from the first phase to one final ranking using a voting policy. The test suite to be prioritized contains *N* test cases that are passed through each individual model. The output of every single model is the order of the test cases in the test suite. That is, each test case has an index representing its position corresponding to each model. Thus, each test case in the test suite will have *M* position values from *M* single models, where *M* is the number of models participating in the ensemble model. We calculate the final weight of a certain test case by adding all the weights of that test case from *M* models. Specifically, in formula (1), a_i is the final weight of the test case i^{th} in the original order, and it is the sum of w_{ji} where w_{ji} represents the weight (position) of the i^{th} test case in the test suite given by the j^{th} model. The test cases are then sorted according to their final weights.

B. Strategies to choose the method for ensemble

In the context of UI testing, many TCP approaches have been proposed. Two common approaches are history-based and coverage-based. Both aim to increase the efficiency of fault detection, yet they use different information and are based on different hypotheses. Moreover, each method carries different advantages. We implement ensemble models to combine those advantages in the hope that they can help increase the overall performance. In detail, we propose three ensemble models, which are History-based, Coverage-based, and History Coverage-based models.

1) History-based ensemble: History-based approach is based on the hypotheses that test cases with errors in the past have a high probability of continuing to detect errors. We choose two methods to include in the history-based ensemble. The first method prioritizes test cases according to the number of failures of each test case in the past. This approach, which is called HBRL hereafter, was proposed by Hemmati et al. [5]. The second is RLTCP [4], which also uses execution history information and a combination of test coverage and reinforcement learning. Although RLTCP is generally more efficient, the first method prevails in the first cycles of the application under test (AUT) because RLTCP needs to go through several cycles to be effective. Combining an ML-based and a traditional method is expected to provide a stable use of the model across stages in software development.

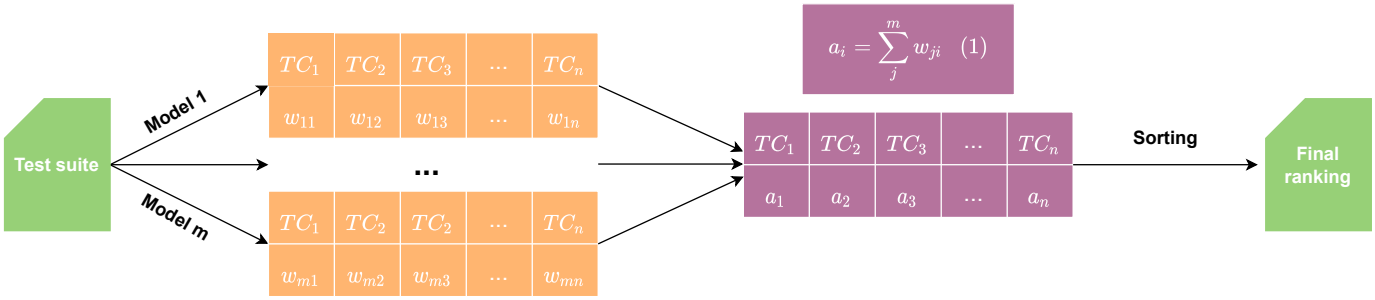


Fig. 1: The basic scheme of the two-step data fusion approach. The first one is to obtain predictive results from different models. The second one is to combine these results to one final result with voting policy

2) Coverage-based ensemble: Coverage-based is based on the coverage information of a test case for one or several components in the AUT. We choose two methods for the coverage-based ensemble. The first method is implemented by sorting based on the number of test steps of a test case, i.e., the length-based method. The test case with the most number of test steps will be ranked first. The second method, called StepGreedy, tries to cover all test steps as quickly as possible using a greedy algorithm. The former considers test cases to be independent of each other, meaning that though test cases include the same test steps, they are considered different. In other words, if two test cases share the same large number of test steps, they are both ranked high even though we just need to execute one of them. Whereas the latter takes into account the relation between test cases, which means that two test steps doing the same action are considered as one test step. After one test case is ranked, its test steps are marked satisfied and removed from further consideration. We select a next test case that maximizes the number of test steps not covered in previously selected tests. In the context of UI testing, certain test cases depend on each other, but there are also test cases entirely independent of each other. Therefore, the expectation of combining these two methods is to increase the efficiency of the model.

3) History Coverage-based approach: We choose RLTCP and StepGreedy from the coverage-based approach for the history coverage-based ensemble. This combination aims to create an approach that uses both execution history and coverage information of test cases.

IV. Experimental Design

A. Overview

We conduct experiments based on how TCP approaches can be used in the software development process. At each iteration or cycle, testers use the models to predict the order of test cases to be executed for the release corresponding to the iteration. They then use the prediction to execute tests that are highly ranked given their time limit.

B. Research Questions

We design the experiments to answer the following research questions:

RQ1: How do the ensemble models perform in comparison with individual models in UI test case prioritization?

We compare the performance of ensemble models with single models to see how effective they are. Single models are the four that we discussed in section III. From that, we find out which ensemble method has the best performance for optimizing early error detection.

RQ2: How does the performance of the ensemble models change over test iterations?

The number of test cases changes after each iteration in software development. Test cases that fail in previous iterations may no longer fail in the current iteration or vice versa. We perform an experiment to represent the performance of ensemble models to compare with the single methods across test execution iterations.

C. Evaluation metrics

We use a standard metric in the Test case selection and prioritization problem called Average Percentage of Fault Detected (APFD). This metric is proposed by Rothermel et al. [19], and it measures the proportion of errors identified at each percentage of test suite execution and then calculates the average to evaluate the effectiveness of a test case prioritizing approach. APFD is calculated according to equation (1). For each fault, the metric determines the first test case index that detects that fault and computes the summation of those indexes of all faults, then divides it by the product of m total faults and n total test cases in the test suite. TFi denotes the number of tests needed to execute before discovering the fault i. That is, if all test cases that expose undetected errors are prioritized and run before test cases that fail to detect new errors, the APFD value for that order will be the highest.

$$APFD = 1 - \frac{TF_1 + TF_2 + TF_3 + ... + TF_m}{mn} + \frac{1}{2n} \quad (1)$$

We use a paired-sample non-parametric Mann-Whitney U test with a confidence level of 0.05 to confirm if the APFD difference between the proposed and other methods is statistically significant. The null hypothesis states that the two methods have no statistically significant difference in APFD. If a paired-sample Mann–Whitney U test has a p-value of less than 0.05, the null hypothesis is rejected, which means

that the difference between two methods in terms of APFD is statistically significant.

D. Data sets

In our experiment, we use four web applications, including Mattermost[1], Moodle[2], Spectrum[3], and Elementary Web[4] as the AUT. The applications are quite popular, adequately mature, and accumulate a considerable number of software iterations.

To create automated UI test cases and run tests, we utilize the test automation tool Katalon Studio[5]. After executing the test suite, the results of test execution will be stored in reports. The report contains the name of test cases, the test steps that each test case includes, the verdict of each test case, and the step that causes failure for test cases. We use these reports as the data sets for validation.

We establish a test suite and tweak its test cases and the AUT's source code to produce mutations across numerous iterations to replicate a realistic software development situation. The specific is described as follows:

- Iteration 1: Initialize the original suite by creating UI test cases based on the original AUT version. The number of test cases in the original suite is at least 20, and some test cases may fail.
- Iteration 2: Test suite will be added at least 10 test cases. Newly added test cases may fail. Test cases that failed in the previous iteration may have been fixed.
- Iteration 3: The suite contains at least 40 test cases. Some components can be added, removed or modified to the AUT to simulate real software development. Some test cases may fail due to these modifications.
- Iteration 4 and after: The number of test cases in each test suite varies, but no test suite has less than 20. The AUT continues to be tweaked with each iteration to simulate software development. Test cases that fail in an iteration can be debugged to ensure that a test case does not repeatedly fail in successive iterations. The last iteration may contain no failing test case.

The details for each data set is represented in Table I. No.test, No.step, No.fail are the number of total test cases, test steps in all test cases, and failing tests respectively that are accumulated across all test suites.

In general, the number of iterations of data sets ranges from 10 to 15. Sixteen groups of senior computer science students correspond to 16 data sets. Each of the four student groups worked independently of one another. This procedure reflects the nature of software development, where functionality and test cases may be added or updated after each release.

[1]https://github.com/mattermost/mattermost-webapp
[2]https://github.com/moodle/moodle
[3]https://github.com/withspectrum/spectrum
[4]https://github.com/vector-im/element-web
[5]https://katalon.com

TABLE I: Number of iterations, test cases, test steps and failing tests in sixteen data sets.

Datasets	No.iteration	No.fail	No.step	No.test
Elementary01	10	104	10,740	690
Spectrum01	10	46	2,041	240
Spectrum02	14	316	3,295	638
Spectrum03	15	80	1,948	431
Moodle01	15	84	9,623	690
Moodle02	15	79	11,796	690
Moodle03	15	96	9,996	690
Moodle04	15	88	7,739	690
Moodle05	15	81	13,368	690
Moodle06	15	92	5,225	675
Moodle07	15	216	11,276	684
Moodle08	15	98	13,995	690
Moodle09	14	161	8,574	640
Mattermost01	11	198	4,631	451
Mattermost02	10	61	8,336	440
Mattermost03	14	183	6,708	637

Fig. 2: APFD of the methods over each iteration for some datasets

V. Results

Table II describes our experimental results of the ensemble methods and single methods across all data sets. Each cell in the table represents the average APFD value of a method on the corresponding data set. The two last rows are the standard deviation and mean values of the APFD scores for each method over 16 data sets.

While EnCov gives a second-worst result at both standard deviation and APFD score with the value of 9.1 and 63.9, respectively. The other two ensemble methods have the highest APFD score and a relatively small variance when compared to individual methods.

Among the based models, StepGreedy is the most stable one with a standard deviation of 4.3. LengthBased has the lowest APFD score of 56.0 and the biggest variance when its standard deviation is significantly higher than other methods (9.1 while the others vary from 4.3 to 6.9). RLTCP often has the highest APFD score among individual methods, and its results are sometimes comparable or even bigger than EnHis

TABLE II: Average APFD over iterations of each method on considered dataset

Datasets	StepGreedy	LengthBased	RLTCP	HBRL	EnCov	EnHis	EnCovHis
Elementary01	69.3	43.2	68.1	66.1	51.7	73.1	**74.4**
Spectrum01	66.8	49.6	67.2	67.7	57.8	70.8	**71.2**
Spectrum02	67.4	44.5	70.2	62.0	55.0	70.3	**74.5**
Spectrum03	63.8	65.0	77.6	70.6	68.0	**77.7**	76.1
Moodle01	63.8	58.5	59.9	54.4	61.5	61.3	**64.4**
Moodle02	64.2	53.5	63.3	54.1	60.2	63.1	**68.5**
Moodle03	64.3	56.7	66.4	65.8	64.4	70.3	**70.5**
Moodle04	64.9	58.9	**75.2**	59.0	67.6	70.9	74.1
Moodle05	66.2	59.9	79.7	57.7	71.3	74.2	**80.5**
Moodle06	66.6	45.2	67.2	60.8	51.2	70.4	**72.4**
Moodle07	67.7	68.3	73.4	65.0	72.2	75.8	**77.5**
Moodle08	68.8	58.5	73.1	62.9	66.6	75.6	**75.8**
Moodle09	73.0	51.0	73.0	61.9	62.8	74.2	**77.6**
Mattermost01	73.6	68.2	74.0	70.6	65.6	74.4	**77.2**
Mattermost02	79.7	70.5	85.3	76.7	74.4	87.6	**88.0**
Mattermost03	68.9	45.3	72.5	66.8	62.5	71.8	**75.0**
Stdev	**4.3**	9.1	6.3	6.1	6.9	5.9	5.3
Avarage	68.1	56.0	71.6	63.9	63.3	72.6	**74.9**

and EnCovHis.

Table III is the p-value of the Mann–Whitney U test mentioned in section IV when comparing EnCovHis with other methods being experimented with within this paper. The bold value indicates the statistically significant difference in APFD between the EnCovHis and the others (p-value ≤ 0.05). The table suggests rejecting the null hypothesis for most of the dataset. The results show that EnCovHis is better than RLTCP on twelve datasets. Even in Elementary01, Moodle07, Mattermost01, their p-values are less than 0.01. On Spectrum01, Spectrum03, Moodle04, Moodle05, p-values are greater than 0.05, although only two of them have average APFDs smaller.

Figure.2 illustrates the APFD score of each algorithm throughout cycles for six data sets. It can be seen that EnCovHis and EnHis may perform badly during some first iterations because of lacking historical information, which is crucial for RLTCP and HBRL to give a good prioritize order, but they get better and become more stable in the latter cycles. Meanwhile, since StepGreedy does not use the execution history, it maintains a small variance during iterations of the data sets.

TABLE III: p-value results of Mann-Whitney U tests between EnCovHis and four single methods

Dataset	StepGreedy	LengthBased	RLTCP	HBRL
Elementary01	**0.004**	**0.001**	**0.001**	**0.005**
Spectrum01	**0.016**	**0.004**	0.131	**0.049**
Spectrum02	**0.007**	**0.001**	**0.015**	**0.002**
Spectrum03	**0.029**	**0.054**	0.736	**0.021**
Moodle01	0.472	0.242	**0.046**	**0.025**
Moodle02	0.145	**0.035**	**0.032**	**0.013**
Moodle03	**0.047**	**0.006**	**0.040**	0.117
Moodle04	**0.007**	**0.009**	0.712	**0.001**
Moodle05	**0.001**	**0.003**	0.484	**0.001**
Moodle06	0.076	**0.002**	**0.030**	**0.009**
Moodle07	**0.002**	**0.021**	**0.005**	**0.003**
Moodle08	**0.013**	**0.003**	**0.040**	**0.001**
Moodle09	**0.033**	**0.001**	**0.033**	**0.002**
Mattermost01	0.120	**0.007**	**0.002**	**0.034**
Mattermost02	**0.038**	**0.038**	**0.038**	**0.021**
Mattermost03	**0.001**	**0.001**	**0.035**	**0.007**

VI. Discussion

This section focuses on answering the research questions in Section IV using the experimental results from Section V.

RQ1: How do the ensemble models perform in comparison with individual models in UI test case prioritization?

As mentioned above, it can be seen that EnCov does not have good results. This may be because the performance of the LengthBased method is too low compared to StepGreedy. Therefore, instead of combining with each other to get a higher result, LengthBased drags the StepGreedy method down, so the APFD scores of EnCov usually lie somewhere between these two methods.

While EvCovHis and EnHis both have promísing results, EnCovHis has a slightly better performance with a more significant average APFD score and a smaller standard deviation. The difference between standard deviations can be explained when examining the standard deviations of every single model. EnHis, which uses two methods with similar standard deviation values, gives a smaller deviation compared to every single method. However, because EnCovHis has StepGreedy, which is the most stable method acting as one of its base models, it has a better standard deviation than that of EnHis.

The Mann-Whitney U test results from Table III suggest that EnCovHis outperforms the four considered existing methods in most of the dataset with the confidence of 95%. However, there are some exceptions, such as Moodle01, Moodle02 where the performance of history-based methods is extremely unstable. The variance of ensemble methods for these datasets is therefore affected, while the Coverage-based method can still keep a good standard deviation, so there is not enough evidence to reject the null hypothesis. RLTCP has a higher overall performance when compared to StepGreedy, LengthBase, and HBRL. Therefore, in some datasets, the APFD score of the ensemble methods only varies around RLTCP's score, and the improvements are not significant.

Examining more closely at the Table II in each dataset,

we can see that the ensemble model will be more likely to have higher results than each of its base models if these base models have a similar performance. Therefore, EnCovHis may perform better than EnHis since its two single models have a closer average APFD score than the two base methods of EnHis.

To sum up, the ensemble method can be used in prioritizing UI test cases to get a higher result if the base models are suitable. The experiment result suggests using methods with similar performances and small variances in order to get higher ensemble performance. The APFD score of the ensemble method throughout cycles depends on its based methods.

RQ2: How does the performance of the ensemble models change over test iterations?

The Fig.2 shows the APFD score of each method for all iterations of a dataset. For most of the cases, the pattern would look similar to that in Spectrum03 and Elementary01. The historical data in the first cycles are not adequate for history-based approaches to make a good decision, so they perform badly at the beginning, then become more stable and exceed coverage-based approaches. Meanwhile, since Step-Greedy does not use the execution history, it maintains a small variance during iterations of the datasets but cannot improve the result after each iteration. EnCovHis and EnHis can neutralize the pros and cons of both History-based and Coverage-based approaches. They suffer less from the lacking of execution data in the first iterations, improve faster for the next cycles while giving a stable and high APFD score in the latter ones.

VII. Conclusion

In this paper, we proposed a test case prioritization technique in regression UI testing by ensembling multiple single models into a single one. We design three ensemble models, which are history-based, coverage-based, and history coverage-based ensembles. These three models were evaluated using 16 data sets with the source code of four different AUT, including Elementary, Spectrum, Moodle, and Mattermost. The evaluating result shows that the history coverage-based model is the best one which achieves the average APFD of 74.9%. This indicates that with suitable based models, the ensemble method can outperform its own base algorithms.

Though ensemble learning is a popular machine learning technique, this is the first time it is applied in UI test case prioritization. Thus, it is a promising direction and can be further explored in the future. Not only in UI testing, but the ensemble method can also be applied in other testing problems. Furthermore, due to the limitation of our data sets, we only conduct experiments based on the idea of the parallel ensemble method, i.e., bagging ensemble. However, other ensemble approaches are still applicable and worth investing in when having larger data and suitable base models.

ACKNOWLEDGEMENTS

This research is funded by Katalon LLC. We would also like to thank students at the University of Science, Vietnam National University, Ho Chi Minh city for participating in our experiments.

References

[1] H. Vocke. (2018) The practical test pyramid. [Online]. Available: https://martinfowler.com/articles/practical-test-pyramid.html?fbclid=IwAR31q-GxAE7fx-8rLO8ayUmeFXsSy6fs1vcqylLmZVtnL2VuWVKWR0I4dboUiTests

[2] R. C. Bryce and A. M. Memon, "Test suite prioritization by interaction coverage," in *Workshop on Domain specific approaches to software test automation: in conjunction with the 6th ESEC/FSE joint meeting*, 2007, pp. 1–7.

[3] D. Di Nardo, N. Alshahwan, L. Briand, and Y. Labiche, "Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system," *Software Testing, Verification and Reliability*, vol. 25, no. 4, pp. 371–396, 2015.

[4] V. Nguyen and B. Le, "Rltcp: A reinforcement learning approach to prioritizing automated user interface tests," *Information and Software Technology*, vol. 136, p. 106574, 2021.

[5] H. Hemmati, Z. Fang, M. V. Mäntylä, and B. Adams, "Prioritizing manual test cases in rapid release environments," *Software Testing, Verification and Reliability*, vol. 27, no. 6, p. e1609, 2017.

[6] R. C. Bryce, S. Sampath, J. B. Pedersen, and S. Manchester, "Test suite prioritization by cost-based combinatorial interaction coverage," *International Journal of System Assurance Engineering and Management*, vol. 2, no. 2, pp. 126–134, 2011.

[7] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software testing, verification and reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[8] J.-M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proceedings of the 24th international conference on software engineering*, 2002, pp. 119–129.

[9] A. Khalilian, M. A. Azgomi, and Y. Fazlalizadeh, "An improved method for test case prioritization by incorporating historical test case data," *Science of Computer Programming*, vol. 78, no. 1, pp. 93–116, 2012.

[10] H. Park, H. Ryu, and J. Baik, "Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing," in *2008 Second International Conference on Secure System Integration and Reliability Improvement*. IEEE, 2008, pp. 39–46.

[11] D. Marijan, A. Gotlieb, and S. Sen, "Test case prioritization for continuous regression testing: An industrial case study," in *2013 IEEE International Conference on Software Maintenance*. IEEE, 2013, pp. 540–543.

[12] Z. Wu, Y. Yang, Z. Li, and R. Zhao, "A time window based reinforcement learning reward for test case prioritization in continuous integration," in *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, 2019, pp. 1–6.

[13] Y. Huang, T. Shu, and Z. Ding, "A learn-to-rank method for model-based regression test case prioritization," *IEEE Access*, vol. 9, pp. 16 365–16 382, 2021.

[14] M. M. Sharma and A. Agrawal, "Test case design and test case prioritization using machine learning," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 1, pp. 2742–2748, 2019.

[15] P. Kaur, P. Bansal, and R. Sibal, "Prioritization of test scenarios derived from uml activity diagram using path complexity," in *Proceedings of the CUBE International Information Technology Conference*, 2012, pp. 355–359.

[16] R. Lachmann, "Machine learning-driven test case prioritization approaches for black-box software testing," in *The European Test and Telemetry Conference, Nuremberg, Germany*, 2018.

[17] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[18] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of computational statistics*. Springer, 2012, pp. 985–1022.

[19] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study," in *Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99).'Software Maintenance for Business Change'(Cat. No. 99CB36360)*. IEEE, 1999, pp. 179–188.

Investigating Cognitive Workload during Comprehension and Application Tasks in Software Testing

Daryl Camilleri
Department of Computer Science
University of Malta
daryl.camilleri.11@um.edu.mt

Chris Porter
Department of Computer Information Systems
University of Malta
chris.porter@um.edu.mt

Mark Micallef
Department of Computer Science
University of Malta
mark.micallef@um.edu.mt

***Abstract* — Software testers are an integral part of software development teams, and consequently need to understand from different perspectives the project entrusted to them. While developers might be required to understand a particular module or area of specialisation within a project, testers' comprehension requirements are more far-reaching [1]. Gaining insights into how testers fare in different comprehension tasks is useful because it sheds light on how we could potentially support the efforts of the testing community. This paper reports the results of a laboratory experiment involving 15 professional software testers. Using NASA Task Load Index as our instrument of choice, we asked participants to carry out eight comprehension and application tasks across four categories (test case design, test automation, bug finding and adequacy analysis). We then analysed the data collected to seek to understand the effect of different task types, education level and participant experience on effectiveness and cognitive workload.**

The results suggest that, while experience is a key element in successful task completion, this is also influenced by task type. In fact, the more experienced persons actually tended to fare worse than their less experienced counterparts in certain tasks (namely, test case design and adequacy analysis). Level of education had no significant bearing on successful task completion but differences in cognitive workload could be observed for both experience and education-level variables.

I. Introduction

Program comprehension is a prerequisite for the effective completion of most tasks in a software engineering context. Sneed [1] argues that the program code is not the only artefact that should be of concern. More specifically, effective members of a development team would need to understand the code, the environment in which it is deployed, the domain which it serves, the stakeholders involved and so on. This is especially the case for software testers, whose comprehension requirements tend to be more demanding than those of a developer. Vanitha and Alagarsamy [2] define software testing as "one of the five main technical activity areas of the software engineering life-cycle that still poses substantial challenges". Other than what seems to be a simple process of checking a sample of runs, software testing encompasses various intricate challenges and enfolds a mixture of activities and techniques.

Indeed, with the constantly growing demand for software and its complexity, ensuring that the software performs as per the required level of quality is becoming highly critical and expensive [3]. From a comprehension perspective, while developers might be required to understand a particular module or area of specialisation within a project, the testers' comprehension requirements usually span a significantly wider area of a project. They also typically deal with a broader range of stakeholders and are expected to carry out a variety of tasks having significant comprehension prerequisites.

In this context, it would be desirable to gain insights into the cognitive workload experienced by testers as they comprehend a task, understand what is required and apply their understanding in completing the task. Such an insight would help guide recruitment, training, work allocation and mentoring efforts within organisations. To this end, we approached this work by posing the following research questions:

- RQ1: How are cognitive load and effectiveness in software testers affected when carrying out different types of testing tasks?
- RQ2: How are cognitive load and effectiveness influenced by an individual's experience and education?

The rest of this paper is organised into five further sections, with Section II providing the necessary background on cognitive workload measurement. Section III outlines the methodology we have adopted in this study. Section IV explores the results and provides the bases for Section V, where the results are discussed in the context of the research questions posed above. Finally, through Section VI, we submit proposals regarding future work, based on the observations made.

II. Cognitive Workload

The term *cognitive workload* (or *mental workload*) is widely used and, while having many definitions, most of them converge on two main aspects: stress and strain [4]. The first refers to the demands of the task, whereas the second refers to its impact on the person carrying it out.

When adapting a definition of mental workload, Galy et al. [5] make reference to Young and Stanton's [6] claim that one should also consider "the amount of attentional resources necessary to perform task as a function of task demand,

DOI reference number: 10.18293/SEKE2022-155

environmental context in which the task is performed, and past experience of individual with task" [5].

A. Measuring Cognitive Workload

The two most widely used instruments for measuring cognitive workload are the subjective workload assessment technique (SWAT) [7] and the NASA-Task Load Index (NASA-TLX) [8]. SWAT measures three dimensions of cognitive workload (time load, mental effort and psychological stress) whereas NASA-TLX has six subscales (mental demand, physical demand, temporal demand, performance, effort and frustration). We have chosen to focus on NASA-TLX because of its wide application across different domains, and its multi-dimensional assessment of workload, which provides a richer insight into the sources of workload over SWAT.

1) NASA-Task Load Index: NASA-TLX [8] is a multi-dimensional scale designed to obtain workload estimates from one or more operators as they are performing a task or immediately afterwards. Since its publication in the 1980s, NASA-TLX has been cited extensively and used in several fields, ranging across nuclear power plant control rooms, certification of aeroplanes, operating rooms, computer-generated fighting, and designing of websites [9]. It consists of a multi-item questionnaire which, when processed, provides an overall task load index with a range between 0 and 100. The higher the rating, the more demanding the task would be. The instrument also provides measurements of six subscales, as indicated above (i.e., mental demand, physical demand, temporal demand, performance, effort and frustration). A weighted load index could also be obtained following a pairwise comparison of these subscales. Like other interface metrics and questionnaires, the TLX application cannot tell what to repair, but it assists research in understanding if variations made to an interface generated a better or deteriorated workload. Although the instrument has most commonly been used in studies that contain physical components [10], existing literature on the topic also includes a substantial amount of work where the study was used to analyse ergonomics in software systems in which the physical component was not necessarily of concern. On the basis of a survey of 500 studies, undertaken 20 years after first being developed [9], its creator noted that while the instrument was originally developed for use in the aviation sector, it had grown to be used in a wide variety of sectors, not least in software engineering.

III. Methodology

In this section, we present the methodology and discuss key decisions taken during its design. All material related to the methodology and results is available on our OSF repository[1].

A. Task Design

Task selection and design was of critical importance in this study. Given the rich spectrum of activities in which software testers are involved, it was important to choose a reasonable subset of tasks that could be carried out in the limited context and time-frame of a lab-based experiment.

1) Tasks Taxonomy: Hrabovská et al. [11] carried out a wide ranging review of software-testing process models. As part of this review, they identified five groups of practices, as follows: planning (21 practices); design (9 practices); set-up (12 practices); execution (13 practices); and monitoring (17 practices). These groups collectively characterise the spectrum of tasks that testers carry out, depending on which process model they follow. We employed this knowledge to guide us in selecting a pragmatic subset of tasks that could be carried out in a lab setting, in a restricted amount of time (approximately one hour). This led us to focus on these four practices: (1) test case design; (2) test automation; (3) exploratory test execution or bug finding; and (4) test adequacy analysis.

In order to minimise participant fatigue, we set out to ensure that the experiment would take approximately one hour. After factoring in an estimated 10 minutes for participant on-boarding and exit interviews, we calculated 40 minutes for data collection. Hence, we deemed it best to design a series of eight tasks, each of which we estimated would take 3-5 minutes to complete. Each task was to be preceded by a 2-minute calming fish-tank video, which enabled participants to reset their mental state in preparation for the task. We also opted to include two practice tasks to be carried out at on-boarding stage, and thus helping to reduce possible participant anxiety due to unfamiliarity during data collection.

We decided to distribute the eight tasks as follows: three *test case design* tasks of increasing difficulty, three *test automation* tasks of increasing difficulty, one *bug-finding* task and one *adequacy analysis* task. During each task, participants were required to read a concise specification or a short snippet of C# code presented on screen. They were then asked to verbally explain how they would complete a specific task related to what they were observing on the screen. For example, after being presented with the specification for a feature, participants were asked to outline how many tests would be required for testing the implementation of that specific feature. The full set of tasks can be found in our replication pack.

B. Data Collection

Data was collected in two ways. Firstly, participants consented to the recording of their onscreen activity and their voice. This enabled us to evaluate, at a later stage, the success rate in the completion of each task. Secondly, participants filled in a NASA-TLX evaluation on paper for each task.

C. Experimental Procedure

Participants were welcomed to the lab, introduced to the experiment and given time to review and sign consent forms. Once the formalities were completed, the participants were introduced to the two practice tasks in order to familiarize themselves with the experiment. At this point, participants iteratively watched a fish-tank video to reset their mental state, carried out a task and completed a NASA-TLX assessment for the task. When all tasks were completed, an exit interview was carried out and the experiment was concluded.

[1]https://osf.io/gnyv7/?view_only=963454403e5c42acbd344d8d8e2c80cd

IV. Results

This section explores the data collected in the experiment guided by the research questions posed in Section I.

A. Participant Demographics

Following initial screening of participants, we selected 20 individuals, of whom 15 made it to the lab and successfully completed the assigned tasks. In terms of experience, 3 participants (20%) had up to 2 years' experience, 8 participants (53%) had between 3 and 5 years of experience, and 4 participants (27%) had 6 years' experience or more. Education levels consisted of 2 participants (13%) having a diploma level of education, 9 participants (60%) holding a first degree, and 4 participants (27%) having postgraduate qualifications. Unfortunately, the gender balance of our cohort was heavily skewed towards male participants, who constituted 14 participants (93%).

B. Task Performance

We post-processed the collected data towards establishing the extent to which participants were successful in their allocated tasks. For each task, we classified the participants' individual performance as *not successful*, *mostly successful* or *successful*. Although determining task success was not a primary goal of this experiment, it provided another dimension from which to evaluate the research questions.

Out of 120 attempts, 24 (20%) were unsuccessful, 44 (37%) were mostly successful and 52 (43%) were successful. Success decreased as tasks became more difficult within each category. Whilst 60% of participants completed the first test design task successfully, this was only the case with 33% in the third task. Similarly, 80% of participants completed the first test automation task successfully, whereas none were successful with the third one. However, is worth noting that while the number of completely unsuccessful candidates increased with each level of difficulty in test case design, the number of unsuccessful attempts at test automation tasks remained constant at 7% (one participant). The bug-finding task had a reasonable level of success, when taking into account that the participants did not know that they were expected to find 10 bugs. Finally, the participants seemed to find test-adequacy analysis the most challenging, with only 27% getting it right and 47% getting it wrong.

1) Effect of Experience on Task Success: We analysed the success by task type and participant experience. The data indicates that experience is a determining factor in task success with 0-2, 3-5 and 6+ year cohorts being successful or mostly successful 92%, 81% and 69% of the time respectively. It is interesting to note that, overall, relatively inexperienced testers had a higher success rate, outperforming individuals more experienced in test case design and test adequacy analysis. We believe that this is due to the recent nature of their formal training. However, experience seems to play a key role in determining success in test automation and bug-finding tasks.

2) Effect of Education on Task Success: When analysing task success by education, the data at hand suggested that the level of education did not have a significant impact on performance. When considering all tasks collectively, the participants having a diploma were successful or mostly successful 81% of the time, participants having a first degree were successful 79% of the time, and those with postgraduate degrees were successful 81% of the time. This contrasts with the more varied success rates when grouping participants by experience.

C. Overall Cognitive Workload

We began our analysis by taking a high-level view of the cognitive workload generated by tasks among our participants. This was done by analysing the distribution of NASA-TLX scores across task types and participants. We did this from the point of view of participant experience and the participants' level of education.

1) Analysing Workload by Experience: When grouping the NASA-TLX scores by task type and participant experience, one notices that the general trend was for cognitive workload to decrease with experience. This was particularly evident in test case design and test adequacy tasks. Both are activities which are taught in all testing curricula, but require repeated practice in order to be applied confidently.

Interestingly, the less proficient testers experienced a lower cognitive workload when carrying out implementation tasks and bug-finding tasks. We believe that this is due to a number of reasons. Firstly, less experienced testers are likely to be fresh graduates, having completed a degree programme focusing on programming skills. Hence, their comfort zone at this point would consist mostly of coding. Secondly, it is probable that experienced testers would specialise in certain subfields of software testing. Therefore, a test engineer specialising primarily in building regression test automation frameworks would be out of touch with bug-hunting skills (and vice-versa). This argument is further strengthened when one notes that the more experienced testers were subject to extreme upper and lower whisker values, which indicate individuals who have specialised in or away from that particular skill.

2) Analysing Workload by Education: When considering all NASA-TLX scores regardless of tasks, one notes that participants with the lowest level of education tended to experience the lightest cognitive workload. The highest score for this cohort was 71, compared to 97 and 73 for undergraduates and postgraduates respectively. This pattern was driven by scores related to *test design*, *test automation* and *bug-finding* task categories, but not *test adequacy analysis*. The measurements for the latter category suggest that higher levels of education result in a lighter cognitive workload when carrying out adequacy analysis. However, it is to be noted that, since there was only one adequacy analysis task and one bug-finding task, the sample plots for these tasks were equivalent to the number of participants in each education group. For instance, the sample of diploma graduates was only 2.

The cohort of participants exposed to the largest cognitive load tended to be first-time graduates, with the top of their interquartile range clearly exceeding 70 for test automation tasks and bug-finding tasks. The resulting mean overall tasks for undergraduates was 52, with 44 for diploma holders and 49 for postgraduates.

D. Individual NASA-TLX Scales

We also examined how participants fared in individual scales of NASA-TLX, the main categories being: (1) mental demand; (2) physical demand; (3) temporal demand; (4) effort; and (5) frustration. It is to be noted that, although we have charted the values for physical demand, we have opted not to analyse this aspect for these tasks. The main reason for this being that it did not offer a scale of interest for our tasks.

1) NASA-TLX Scales by Experience: Beginning with the scales related for test case design tasks (Figure 1(a)), we have noted that the less proficient participants experienced the heaviest cognitive load in all the scales, with notable peaks for mental demand, frustration and effort. This is interesting when considering that this group accomplished these tasks at optimal levels, outperforming the other two cohorts. It is also interesting to note that their performance score was higher than that of the other cohorts, indicating that they did not feel they were successful with the task.

In the test automation tasks (Figure 1(b)), mental demand, frustration, effort and performance were quite similar for all three groups, with the least experienced group recording the lowest values. As regards the temporal demand value, there was a marked spike among the participants with 3-5 years' experience.

The radar chart for bug finding (Figure 1(c)) indicates elevated levels of effort for both the group with 0-2 years' experience and the group having 3-5 years' experience. The more experienced testers in the group seem to have been minimally affected in all scales except mental demand. Moreover, on the basis of the compiled data, we established that the most experienced cohort performed exceptionally well in this task.

Finally, following an analysis of test adequacy analysis (Figure 1(d)), we observed a somewhat similar picture to test case design. More precisely, the least experienced cohort registered the highest levels of scales, compared to other cohorts, even obtaining an average score of 75 for effort. This effort paid off, with the group significantly outperforming the others in test adequacy analysis. It is worth nothing that all groups scored a very low score on the performance scale, which would suggest that participants felt they were successful in this particular task.

2) NASA-TLX Scales by Education: When analysing individual NASA-TLX scales from the perspective of the participants' education, one of the most notable values is the level of frustration experienced by diploma-level holders when carrying out the test adequacy analysis task. Being the most pronounced component in this task it called for particular attention, when one considers that all other scales for the same task scored similar values to other cohorts. This outcome may be due to a lack of training in this particular technique at diploma level.

One also notes that first-degree graduates tended to be exposed more than other cohorts, scoring an average of 60 in every task category except test case design. Postgraduates had similar temporal demand readings in test automation and bug-finding tasks. Unlike the other two groups, diploma holders tended rarely to experience any significant temporal demand.

V. Discussion

This section seeks to address each of the two research questions defined in Section I, discussing the respective outcomes on the basis of the results presented in Section IV.

A. Effect of Types of Testing Task (RQ1)

The discussion of RQ1 revolves around effectiveness and cognitive load.

1) Task Effectiveness: The results indicate that participant effectiveness decreased when we increased the difficulty level of tasks in both the test design category and the test automation category. However, the pattern of diminished effectiveness differed substantially between the two categories. In test design tasks, the number of participants who were completely unsuccessful in their attempts, increased from 7% to 27% to 40% for each successive task and difficulty increment. More encouragingly, the percentage for the test automation category remained constant at 7%. A closer look at our raw data revealed that the failing participant was a different person in each test automation task, leading us to conclude that the failure may be due to lack of familiarity with the specific test automation technique being used in that task. In contrast, the participants who failed the second test design task, also failed the third one, and were joined by three new participants who were similarly unsuccessful in the task. This suggests that test design tasks tend to be more cohesive in nature than their automation counterparts.

The number of partially successful candidates in the test design tasks remained relatively constant from one task to the other (33%, 27%, 27%). We have interpreted this to suggest that, with test case design, participants either know a technique or they do not, with little room for a middle ground. On the other hand, test automation tasks saw the partially successful range going from 13% to 47% to 93%, but with no candidates completing the task successfully. This would suggest that there are multiple ways in which to carry out the same test automation task and that the nature of test automation would allow a wider margin of error without resulting in complete failure.

The bug-finding task was not based on any specific technique but relied on the participants' level of observation and ability to detect anomalies. More than half the participants (53%) managed to find all 10 bugs on the screenshot, while 27% found at least 8 (which was our boundary for a partially successful rating).

Finally, participants found test adequacy analysis, the most challenging task of all with 47% failing the task completely and only 27% completing it successfully.

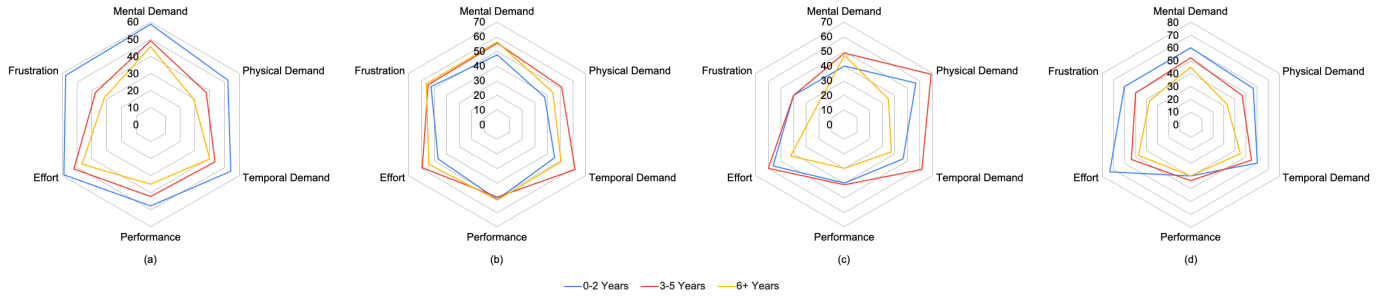


Fig. 1. NASA-TLX scales by experience for (a) test design, (b) test automation, (c) bug finding, and (d) adequacy analysis

2) Cognitive Workload: When analysing the NASA-TLX scores across all participants, we observed that both the test design tasks and the bug-finding task had a mean score of 47. However, there were differences in the distribution of the scores. Whereas interquartile scores (middle 50% of participants) for test design tasks were compacted between 34 and 60, the interquartile range for bug finding ranged from 26 to 74. This suggests that test design tasks generate a more consistent cognitive load than bug finding, which tends to be more varied. Test automation tasks generated a mean cognitive load of 55, with an interquartile range of 39 to 70. Finally, test adequacy analysis generated a mean load of 52 with an interquartile range of 40 to 63. This makes test adequacy the most compactly distributed task category.

In the NASA-TLX scales (see Table I) the type of task had minimal effect on mean mental demand, temporal demand and effort. The exception to this was that the test design had a significantly lower value (45) than the cluster of the other three categories (56, 52, 53). However, performance and frustration were significantly affected by the type of task. The performance scale indicated that participants were most confident with test design and bug finding, but less so with adequacy analysis and test automation. It is worth noting that the maximum value of 50 was nowhere near the higher end of the scale's bounds, thus indicating that the participants were relatively confident in their performance, even if the actual results appeared to point in a different direction.

Frustration also exhibited a certain variability based on the type of task being carried out. Participants found test automation to be the most frustrating category with a score of 54. This was followed by adequacy analysis (49), test design (40) and bug finding (35).

B. Effect of Experience and Education (RQ2)

As discussed in Section IV, the participants' experience in the field had an impact on both their effectiveness and cognitive workload. However, whilst education did demonstrate some variability in cognitive workloads, it had a negligible impact on task effectiveness.

1) Task Effectiveness: The idea that experience would have an effect on task effectiveness was arguably an expected outcome of this work. However, we were surprised to observe that this impact was not always positive with the less experienced participants outperforming more experienced ones in test design and adequacy analysis tasks. This may be due to a combination of two factors. Firstly, less experienced candidates would have just recently been trained in the methodological aspects of testing required by these two types of tasks. Secondly, in our conversations with the participants collectively working across a spectrum of studies over these past years, we have observed that testers tend to eventually settle into a preferred role or specialisation. For example, one might specialise as a test engineer, a test analyst or a test lead. Each of these specialisations would result in certain skills being given less attention in favour of others, over time.

The opposite held true for test automation and bug finding. An interesting point, here, was that the increased effectiveness in test automation is not impressive, in that it ranged from 89% to 92% to 100% as the level of experience increased. One could argue that an 89% success rate is actually to be expected. However, the differences in bug finding were much more pronounced, ranging from 67% to 79% to 100%. We believe that bug finding is one skill that benefits more from a trained eye, developed through experience and practice, as opposed to a technique that could be applied methodologically.

2) Cognitive Workload: The highest mean levels of cognitive workload were exhibited in test automation tasks for candidates with 3-5 years of experience (57) and undergraduates (59). The interquartile range for 3-5 years' experience ranged from 40 to 75, whereas that of the undergraduate cohort ranged from 41 to 78. The means were at 63 and 57 respectively, indicating that experience produced a wider distribution of NASA-TLX scores. In both cases, the scores seemed to be driven by all subscales concurrently, with no specific subscale providing a disproportional influence.

As regards test design tasks, cognitive load decreased as experience increased, going from a mean of 57 (0-2 years) to 47 (3-5 years) to 41 (6+ years). The influence of education in this category was less pronounced and moved in the opposite direction, with means of 43, 46, and 51 for diploma holders, first-degree holders and postgraduates respectively.

Bug finding inflicted the least cognitive demands on both the least educated (34) and the least experienced (42). Among the least-qualified participants, this was driven by low levels of mental demand, frustration, effort and temporal demand, whereas in the least experienced the higher mean was driven

TABLE I
MEAN VALUES FOR NASA-TLX SCALES

	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration
Test Design	50	38	45	31	52	40
Test Automation	54	47	56	50	55	54
Bug Finding	47	57	52	38	55	35
Adequacy Analysis	52	45	53	41	56	49

by higher levels of effort (57 vs 35) and temporal demand (47 vs 30). This suggested that a lack of experience generates the need for more effort and concentration than does a lack of education.

C. Threats to Validity

This work is subject to the same threat to external validity as other experiments, in that it is a single experiment. Although we have presented some of the results on the basis of empirical analysis, we cannot claim that these results are representative of the whole testing population. Nevertheless, we are confident that they provide a useful insight and form a foundation for further study. We also mitigated internal validity risks through rigorous experimental procedure and utilising NASA-TLX, which has been used successfully in countless studies.

VI. CONCLUSION AND FUTURE WORK

In this paper, we set out to shed light on the cognitive workload experienced by testers from different cohorts as they attempted to complete a range of tasks typical of the field. Our results uncovered interesting patterns in effectiveness based on the type of task alone. They also indicated that, although experience is a key influence on successful task completion, success is also conditioned by task type. Moreover, more experienced persons tended to fare worse than their less experienced counterparts in certain tasks (test case design and adequacy analysis). Level of education had no significant bearing on successful task completion but differences in cognitive workload could be observed for both experience and education level variables. Here too, it was experience that exerted the strongest influence.

Throughout the course of analysing our data and writing the paper, we have identified a number of shortcomings, which would be addressed as part of our future work.

A. Future Work

This study lays the foundations for a number of opportunities for further exploration. Firstly, it would be useful to observe a better balance of demographic properties, such as a much wider representation of the female population. Moreover, a more balanced sample of education level and experience would be similarly highly desirable. More varied cohorts would shed light on whether the results presented here do indeed hold for a wider population. In addition, it would be beneficial to refine the experimental protocol to balance out the number of tasks within each category and provide the space for more qualitative data through follow-up discussions. This would make it possible to elaborate upon mere numbers, and gain deeper insight into, for example, why the more experienced persons tended to fare worse than their less experienced counterparts in certain tasks.

Once the data would have been sufficiently replicated, we would be in a better position to apply our observations to producing guidelines for companies regarding the management of software testers. At present, the data presented here could be used to inform recruitment decisions, team composition decisions, project management, training paths and promotion ladders in the field of software testing.

REFERENCES

[1] H. Sneed, "Program comprehension for the purpose of testing," in *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.*, 2004, pp. 162–171.

[2] A. Vanitha and K. Alagarsamy, "Software testing in cloud platform: A survey," 04 2019.

[3] A. Bertolino, "Software testing research: Achievements, challenges, dreams," 06 2007, pp. 85 – 103.

[4] M. S. Young, K. A. Brookhuis, C. D. Wickens, and P. A. Hancock, "State of science: mental workload in ergonomics," *Ergonomics*, vol. 58, no. 1, pp. 1–17, 2015.

[5] E. Galy, J. Paxion, and C. Berthelon, "Measuring mental workload with the nasa-tlx needs to examine each dimension rather than relying on the global score: an example with driving," *Ergonomics*, vol. 61, no. 4, pp. 517–527, 2018.

[6] J. Q. Young, R. M. Wachter, O. Ten Cate, P. S. O'Sullivan, and D. M. Irby, "Advancing the next generation of handover research and practice with cognitive load theory," *BMJ quality & safety*, vol. 25, no. 2, pp. 66–70, 2016.

[7] G. B. Reid and T. E. Nygren, "The subjective workload assessment technique: A scaling procedure for measuring mental workload," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 185–218.

[8] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[9] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, no. 9, pp. 904–908, 2006. [Online]. Available: https://doi.org/10.1177/154193120605000909

[10] H. Mansikka, K. Virtanen, and D. Harris, "Comparison of nasa-tlx scale, modified cooper–harper scale and mean inter-beat interval as measures of pilot mental workload during simulated flight tasks," *Ergonomics*, vol. 62, no. 2, pp. 246–254, 2019, pMID: 29708054. [Online]. Available: https://doi.org/10.1080/00140139.2018.1471159

[11] K. Hrabovská, B. Rossi, and T. Pitner, "Software testing process models benefits & drawbacks: a systematic literature review," *arXiv preprint arXiv:1901.01450*, 2019.

Visualization of automated program repair focusing on suspiciousness values

Naoki Tane, Yusaku Ito, Hironori Washizaki, Yoshiaki Fukazawa
Department of Fundamental Science and Engineering
Waseda University
Tokyo, Japan
n.tane0228@fuji.waseda.jp

Abstract*–Automated program repair (APR) can realize efficient debugging in software development. Automated program corrections using genetic algorithms (GA) can repair programs, including those with multiple bugs, but the repair process of GA-based APR is difficult to understand using logs because many modification program codes are generated. Consequently, Matsumoto et al. implemented a methodology for visualizing the process. Their proposed methodology provides an intuitive understanding of the conformance values (test case pass rates), generations, states, and operations performed to generate each variant; however, it lacks sufficient information to analyze whether defect localization is appropriate in APR. Herein we propose a new methodology to visualize the impact of fault localization on program evolution in GA-based APR and create a new tool. Additionally, a case study demonstrates the effectiveness of the proposed methodology and future works are considered.

Keywords*–Visualization; Genetic Algorithm; Automated Program Repair; Fault Localization; Bug Localization

1 Introduction

Automated program repair (APR) is a technique that removes bugs without human intervention. APR outputs a bug-free program when given a buggy program and test suites.

kGenProg, which is a Java programmatic implementation of genProg, is a tool that uses genetic algorithms (GAs) for APR (GA-based APR)[1]. A key feature of kGenProg is its high portability. Users can easily change its parameters because kGenProg has an adaptable fault localization framework. However, it is difficult to analyze the repair process using the logs alone because kGenProg generates multiple programs during the modification process.

Macaw is an open-source software to visualize the evolutionary process of programs by kGenProg[2]. Macaw shows the code genealogy as a bird's eye view tree structure and detailed variant information, where a variant is a program generated in the evolutionary process. Its visualization provides an intuitive understanding of each variant. This information can be used to adjust the kGenProg parameters. One shortcoming is that Macaw does not provide enough information to analyze whether the fault localization is appropriate.

To address this issue, we propose a new methodology to visualize the impact of fault localization on the evolution of programs in kGenProg. Specifically, we create a new tool called Grackle based on this methodology and evaluate the visualization efficiency via a case study.

2 Background and problem

2.1 Fault Localization

Fault localization methods identify faulty program lines based on information obtained from the success or failure of test cases. A common category is spectral-based methods[3]. Spectral-based methods assign suspiciousness values according to a program statement's likelihood of a flaw. Tarantula[4], Ample[6], Jaccard[5], Ochiai[5], and Zoltar[7] represent Spectrum-Based Fault Localization(SBFL) to calculate the suspiciousness values (Table 1). The effectiveness depends on the given test case and program.

DOI reference number: 10.18293/SEKE2022-159

Table 1. Formulas to calculate the suspiciousness values for select SBFL methods

SBFL	formula for calculating the suspiciousness value
Ample	$Suspiciousness = \left\lvert \frac{a_{ef}}{a_{ef}+a_{nf}} - \frac{a_{ep}}{a_{ep}+a_{np}} \right\rvert$
Jaccard	$Suspiciousness = \frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}}$
Ochiai	$Suspiciousness = \frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})\times(a_{ef}+a_{ep})}}$
Tarantula	$Suspiciousness = \frac{\frac{a_{ef}}{a_{ef}+a_{nf}}}{\frac{a_{ep}}{a_{ep}+a_{np}}+\frac{a_{ef}}{a_{ef}+a_{nf}}}$
Zoltar	$Suspiciousness = \frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}+10000\times a_{nf}\times\frac{a_{ep}}{a_{ef}}}$

a_{ep}: Number of successful tests that executed the line
a_{ef}: Number of failed tests that executed the line
a_{np}: Number of successful tests that did not execute the line
a_{nf}: Number of failed tests that did not execute the line

2.2 APR

Many researchers have actively investigated technologies to increase the efficiency of debugging because debugging accounts for half of system implementation and testing costs[8]. These studies often focus on APR. APR removes bugs from buggy programs without human intervention[9]. It works by taking a buggy program and test cases as input and outputs a bug-free program.

2.3 kGenProg

GenProg is an APR methodology based on source code reuse[10]. kGenProg is an APR tool written in Java. It is a reimplementation of GenProg, which automatically repairs bugs using GAs. kGenProg works as follows. First, it infers the bug's line using a fault identification technique. Second, it generates multiple variants by modifying the lines that contain bugs. This step has two primary operations. One is mutation. Mutation means making minor changes to a selected variant to create a new variant such as insert, delete, or replace. The other is crossover. Crossover mixes two selected variants to generate a new variant. Third, unit tests are run on the generated variants. A repaired program is outputted if a variant passes all the tests. If not, kGenProg selects some of the generated variants and generates new variants based on them. Selection means that kGenProg takes some variants from the latest generation. kGenProg decides which variants to select by the pass rate of the unit test. Additionally, it selects some variants from the previous generation to account for the possibility of a poor pass rate for all generated variants.

2.4 Problem in use and existing visualization

Macaw visualizes the evolutionary process of programs with kGenProg. It was developed by Matsumoto et al.. Macaw's visualization provides an intuitive understanding of each variant's fitness value (test case pass rate), generation, state, and operation (insertion, deletion, replacement, crossover, or copy). Macaw helps adjust the parameters of the APR. However, it does not provide sufficient information to comprehend the impact of failure localization on the APR in the program's evolution. To address this shortcoming, we propose a new methodology to visualize the impact of fault localization on program evolution in kGenProg and evaluate the effectiveness of our methodology. This study aims to answer the following two research questions (RQs):

RQ1: Does the proposed methodology facilitate the understanding of fault localization?

RQ2: Is the effect of fault localization in GA-based APR easily understood?

3 Visualization of code genealogy with fault-localization results

Similar to the Macaw project, our visualization displays the evolutionary process of kGenProg with the code genealogy shown as a bird's eye view tree structure and detailed variant information. These two factors can explain the representation of the alleged values. The color intensity represents suspicious values set for the row that operated to generate each variant. The color intensity also represents each row that is subject to the suspicious values of each variant. Figure 1 depicts the visualization of the suspiciousness values in the detailed information of the variant. Our visualization method provides an intuitive recognition of the suspiciousness value for the lines in the bug framework. The source code of each variant is displayed at the top of the detailed information, and the suspiciousness is indicated in red. The darker the red color, the higher the suspiciousness value in the line.

Figure 2, which shows the flow to generate child variants, overviews the proposed visualization of the suspiciousness values for the code genealogy. First, fault localization calculates the suspiciousness values of each line of the parent variant. This is similar to how kGenProg generates variants. Second, the suspiciousness value set at the line where kGenProg is performed is identified by the color intensity of the edge of the tree structure in the bird's eye view.

Figure 3 shows the visualization of the suspiciousness value for the code genealogy. Similar to the visualization proposed by Macaw, each node represents a single variant,

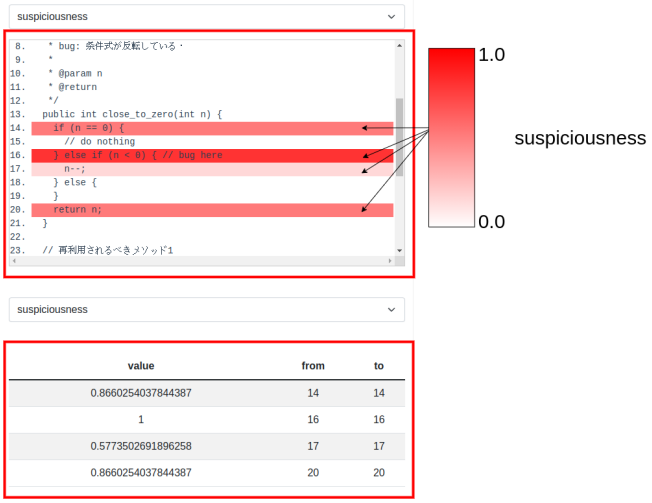


Figure 1. Proposed method to visualize the suspiciousness values in the detailed information of the variant

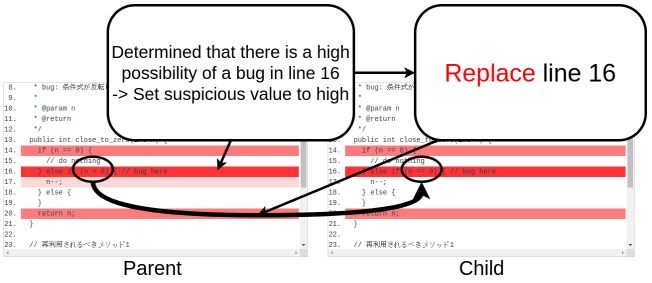


Figure 2. Flow of the generation of child variants

and nodes on the same y-axis mean that they are from the same generation of kGenProg. Circle nodes represent newly generated variants, and small circle nodes denote variants copied from all ages. Crosses indicate variants that failed to compile or invalid variants. The proposed visualization describes the number of nodes in each generation below the cross. The adaptive value is an indicator of the pass rate. In our method, the darker the green color, the higher the adaptive value.

Unlike Macaw, the red intensity of the edge of the tree structure in the bird's eye view of our method represents the suspiciousness value set at the point where kGenProg operated (insertion, deletion, or replacement). The darker the color, the higher the suspiciousness value of the changed part. It should be noted that edges representing crossover and copy operations are not colored because the crossover operation combines codes of two variants and the copy operation in kGenProg does not make any changes. Hence, areas with high suspiciousness values are easily visualized and the influence of fault localization on APR can be understood intuitively. Similar to Macaw, clicking on a node shows details of the generated variants.

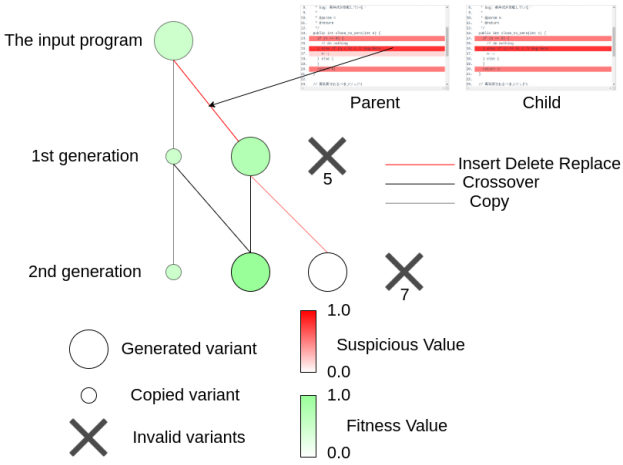


Figure 3. Visualization of the suspiciousness value for the code genealogy

4 Tool implementation

To implement our methodology, kGenProg must be modified. By default, kGenProg has an option to output JSON files. In addition to JSON files, the source code and suspiciousness values must also be outputted in JSON to implement our method. Therefore, we created a modified kGenProg to output the source code and suspiciousness values as JSON files. We also created a tool to implement our

methodology. We call this tool Grackle. Grackle can visualize the suspiciousness values by reading the JSON file output from the modified kGenProg.

Figure 4 shows the flow using Grackle. First, our methodology builds a modified kGenProg, which outputs source code and suspiciousness values as JSON files and creates an executable file (JAR file). Second, our methodology runs the auto-fix in the executable file. Finally, Grackle is used to visualize the flow of the automated modification by inputting the JSON file output from the involuntary conversion.

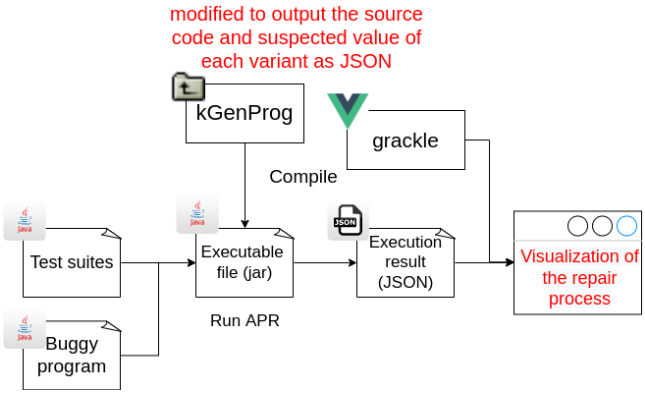


Figure 4. Flow of Grackle

Figure 5 shows a screenshot of the code genealogy visualized by Grackle. The left side shows the code genealogy, while the right displays detailed variant information. Grackle highlights the selected variant with a blue border and the parent variant with a light blue frame. Hence, users can visualize the suspiciousness value of each line of code in the variant. Additionally, the red intensity of the edge of the tree structure in the bird's eye view represents the suspiciousness value set at the changed line in the code system.

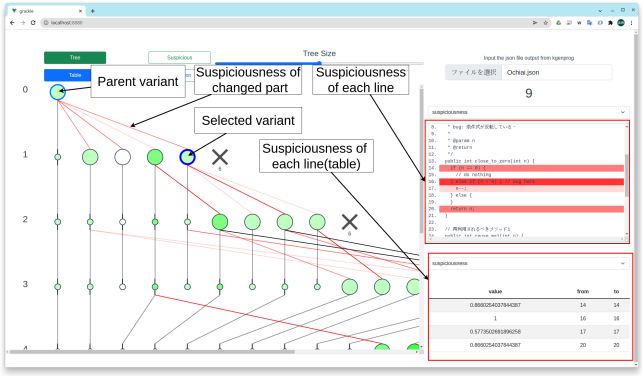


Figure 5. Screenshot of the code genealogy visualized by Grackle

The source code differences and test results in Variant Details can be viewed by changing the mode. Hence, the edge types in Code Genealogy and Macaw can express various operations. Figure 6 shows an example of the visualization when changing the mode in Grackle. Figure 7 shows a screenshot.

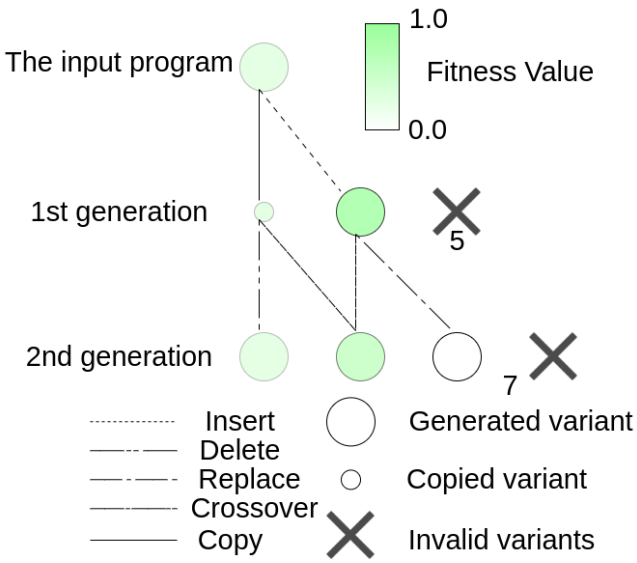


Figure 6. Example of the visualization when changing the mode in Grackle

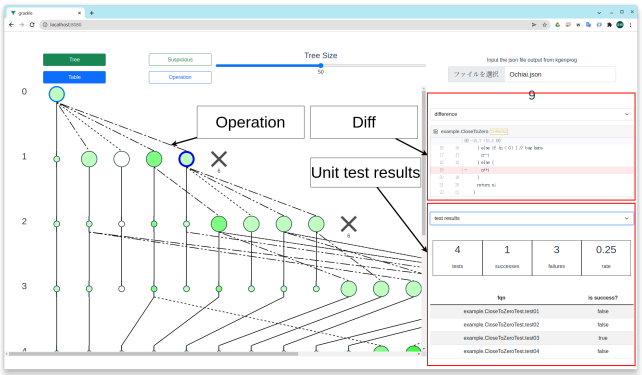


Figure 7. Screenshot of the code genealogy visualized by Grackle

We implemented Grackle using the Javascript framework Vue.js. Grackle can also run in modern web browsers such as Google Chrome, Firefox, Safari, and Microsoft Edge. This example handles a small log. A normal behavior is observed even for a relatively large record with 1970 variants and a generation number of 100. However, it does not support visualizations of the suspiciousness values for mul-

tiple java programs. This support should be addressed in the future. Clicking on a node in the bird's-eye view on the left shows detailed information about the variant on the right. Clicking on the button in the upper left corner changes the edges of the bird's eye view. The correct upper dialog allows users to select a JSON file representing the APR process output by kGenProg. Then users can select the code and the Diff representing the alleged value using the select box above the detailed information. They can also select a table of suspiciousness values and unit test results using the selection box below the detailed information.

5 Case study

5.1 Conditions

We performed an APR with kGenProg under the following conditions to verify whether our methodology is effectively implemented:

Program: kGenProg's QuickSort test program (QuickSort.java)
Fault Localization: Ochiai

We also examined the RQs by comparing Macaw and Grackle visualizations for APR under the above conditions. Macaw is a project to visualize the evolution process of programs using kGenProg, while Grackle is the tool devised in this study.

5.2 Results

Figure 8 shows a screenshot of Macaw. Figure 9 shows a screenshot of Grackle. Both Macaw and Grackle provide code genealogy to intuitively understand the test case pass rate, generation, status, and operation performed by kGenProg to generate each variant. Both also generate detailed information about the variants, allowing users to understand the differences between the before and after operations and the unit test results. In addition, Grackle provides the alleged value of the changes made. Grackle also shows the suspiciousness value calculated for each variant line by fault localization in the variant details.

5.3 RQ1: Does the proposed methodology facilitate the understanding of fault localization?

Macaw does not give information about the suspiciousness with bugs calculated by fault localization in each variant. On the other hand, Grackle provides an intuitive understanding of which line of code in each variant is responsible for the calculated suspiciousness value. Thus, the proposed method facilitates the understanding of fault localization.

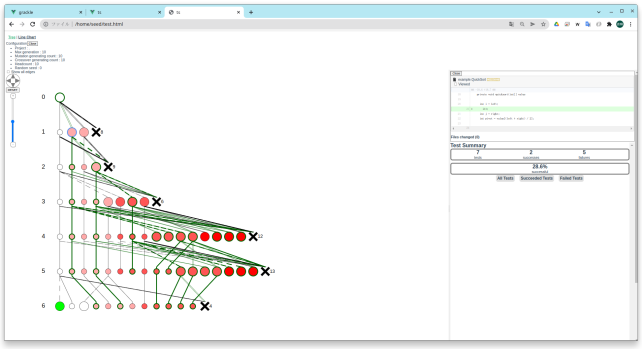


Figure 8. Screenshot of Macaw

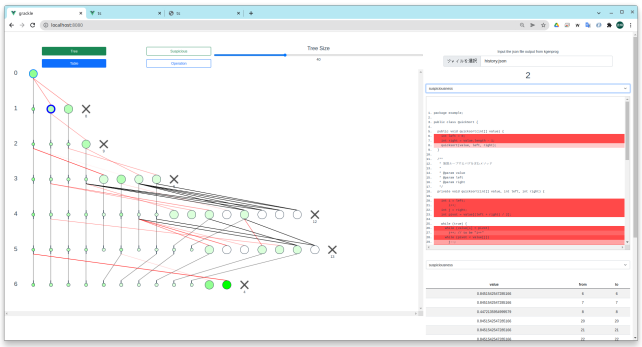


Figure 9. Screenshot of Grackle

5.4 RQ2: Is the effect of fault localization in GA-based APR easily understood?

Macaw's code genealogy does not provide information about the alleged value of the point of change in each operation. On the other hand, Grackle's code genealogy generates information about the alleged value of the point to be changed in each mutation operation. This information helps realize an intuitive understanding of APR's fault localization behavior using GAs. Thus, the proposed method clarifies the effect of fault localization.

6 Conclusion and future work

Herein we propose a methodology and tool to visualize the impact of fault localization on the evolution of programs in kGenProg. The effectiveness was evaluated via a case study. The case study qualitatively demonstrates the usefulness of our methodology and tool.

We visualize three prominent use cases: to compare different fault localization strategies, to select an appropriate fault localization strategy, and to realize an intuitive understanding of changes made by KGenProg. kGenProg can easily change fault localization strategies. The proposed methodology supports an intuitive understanding of

the changed parts of the code using the code genealogy and the suspiciousness values. For example, using a test suite and a buggy program as inputs, different fault localization strategies (Ample, Jaccard, Ochiai, Tarantula, or Zoltar) can be applied and the APR run. The proposed methodology can compare fault localization strategies directly and elucidate the impact of automated corrections by strategy.

The proposed methodology can provide an understanding for setting suspiciousness values for fault localization. It can be employed to check the code genealogy and the details of the variant to verify that the suspiciousness value is set appropriately. For example, a fault localization strategy may negatively impact the APR if the suspiciousness value is set low for a buggy part or high for a non-buggy part. In this way, fault localization can be evaluated, allowing users to select an appropriate bug identification tool. This should improve the efficiency of APR.

Use cases assume that the fault localization strategy is working properly. Our methodology highlights significant changes made by kGenProg. The visualization of suspiciousness values using code genealogy edges shows changes with high suspiciousness values (i.e., where bugs are likely to be present). In this case, a bug is likely to be fixed accurately.

In the future, we plan to evaluate our methodology and tool quantitatively. We also plan to devise use cases for the proposed methodology and verify the practicality of our methodology and tool.

References

[1] Y. Higo, S. Matsumoto, R. Arima, A. Tanikado, K. Naitou, J. Matsumoto, Y. Tomida, and S. Kusumoto, "kGenProg: A High-Performance, High-Extensibility and High-Portability APR System," 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 2018, pp. 697-698.

[2] Y. Tomida, Y. Higo, S. Matsumoto and S. Kusumoto, "Visualizing Code Genealogy: How Code is Evolutionarily Fixed in Program Repair?," 2019 Working Conference on Software Visualization (VISSOFT), 2019, pp. 23-27, doi: 10.1109/VISSOFT.2019.00011.

[3] J. Xuan and M. Monperrus "Learning to Combine Multiple Ranking Metrics for Fault Localizetion" 2014 IEEE International Conference on Software Maintenance and Evolution pp. 191-200 2014.

[4] J. A. Jones, M. J. Harrold and J. Stasko, "Visualization of test information to assist fault localization", Proceedings of the 24th international conference on Software engineering, pp. 467-477, 2002.

[5] R. Abreu, P. Zoeteweij and A. J. Van Gemund, "On the accuracy of spectrum-based fault localization", Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION 2007. TAICPART-MUTATION 2007, pp. 89-98, 2007.

[6] V. Dallmeier, C. Lindig and A. Zeller, "Lightweight bug localization with ample", Proceedings of the sixth international symposium on Automated analysis-driven debugging, pp. 99-104, 2005.

[7] T. Janssen R. Abreu and A. J. C. van Gemund "ZOLTAR: A toolset for automatic fault localization" Proceedings of the International Conference on Automated Software Engineering (ASE'09) - Tool Demonstrations.

[8] Britton, T., Jeng, L., Carver, G., Cheak, P. and Katzenellenbogen, T. (2013). Reversible Debugging Software: Quantify the time and cost saved using reversible debuggers.

[9] YASUDA, Kazuya, ITOH, Shinji, NAKAMURA, Tomonori, HARADA, Masao, HIGO, Yoshiki. Automated Program Repair Using Donor Code Generation Based on Features of Targeted Systems. Computer Software. 2021, vol. 38, no. 4, p. 4_23-4_32.

[10] C. Le Goues, M. Dewey-Vogt, S. Forrest and W. Weimer, "A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for $8 Each", ICSE'12, pp. 3-13.

Beyond Numerical – MIXATON for outlier explanation on mixed-type data

Jakob Nonnenmacher
University of Oldenburg
Oldenburg, Germany
jakob.nonnenmacher@uol.de

Jorge Marx Gómez
University of Oldenburg
Oldenburg, Germany
jorge.marx.gomez@uol.de

Abstract— Outlier explanation approaches are employed to support analysts in investigating outliers, especially those detected by methods which are not intuitively interpretable such as deep learning or ensemble approaches. There have been several studies on outlier explanation in the last years. Nonetheless, there have been no outlier explanation approaches for mixed-type data. In this paper we propose multiple approaches for outlier explanation on mixed-type data. We benchmark them by using synthetic outlier datasets and by generating ground-truth explanation for real-world outlier datasets. The results on the various datasets show that while there is no approach that dominates others for all types of outliers and datasets, some can offer a consistently high performance.

Keywords-outlying aspect mining; outlier explanation; mixed-type; anomaly detection

I. INTRODUCTION

Being able to understand the output of a machine learning model is a core requirement for its successful deployment in real-world applications. Outlier detection models do not always fulfill this requirement. To remedy this, a number of outlier explanation methods have been proposed in the last years [1–5]. Outlier explanation is important because outlier detection methods are often used in a more explorative context which makes it important for analysts to investigate the results before they can take action. Furthermore, outlier detection methods often only provide information on whether something is an outlier but not why it is an outlier [1, 2, 5]. More recent state-of-the-art approaches based on deep-learning or ensemble methods are especially known for not being easily explainable [6]. This includes methods like IForest [7] as well as autoencoders [8]. Another aspect, which makes outlier explanation relevant, is the fact that all outlier detection methods employ some kind of statistical measure to determine the outliers. The problem with this is that the statistical measure might select an entry as an outlier that does not necessarily match the outlier definition within the domain the detector is used [4]. To weed out these potential false positives from the results, the analyst has to understand why an entry has been selected as an outlier.

Despite the large number of outlier explanation methods that have been proposed so far, most of them have been created for numerical data. A few methods explicitly address categorical data [9, 10] and one method has been presented which claims to work for both numerical as well as mixed-type data but has only been evaluated on the former [2]. Data in real-world applications is often mixed-type and multiple outlier detection methods and adaptions for mixed-type data exist [11–13]. Finding outliers, including those that are only apparent when considering both numerical and categorical features together, is important for uncovering fraud or identifying cyber-attacks. The fact that no outlier explanations approaches exist for mixed-type data makes the application of state-of-the-art mixed-type outlier detection approaches potentially less effective and practical for these and other real-world applications. To address this challenge, we propose and evaluate multiple variations of outlier explanation approaches for outliers detected on mixed-type data. Overall, the contribution of this work includes:

- Multiple adaptions of outlier explanation approaches for mixed-type data
- Creating a benchmark for outlier explanation approaches for mixed-type data
- Recommendations for which methods to use for mixed-type data

The rest of this paper is organized as follows: Sec. II introduces related work. Sec. III presents our developed approaches. Sec. IV contains the evaluation as well as the discussion of our results and in Sec. V we conclude our work.

II. RELATED WORK

The first prevalent outlier explanation approach is the so-called score-and-search approach in which subsets of the original full featurespace are created and then scored using a scoring metric. The subset featurespace in which an outlier receives the highest score is selected as the explaining subspace for that outlier. Multiple variations of this score-and-search approach have been proposed, focusing on reducing the number of subspaces to score [3, 4, 14–16] and on making the scores comparable between subspaces of different dimensionality [3, 15, 16].

The second prevalent approach is the so-called feature transformation and importance approach. Most of these approaches work by using a classifier that is trained to differentiate between the inliers and the outliers to explain. Then a feature selection technique like Lasso is used to determine the importance of the individual features for this classification task [5, 17–19]. One technique that is similar to this approach is to train a classification model and to then extract an explanation using the general model explanation technique SHAP which was

DOI reference number: 10.18293/SEKE2022-110

presented by [20]. SHAP is a method derived from game theory and attributes credit to each feature for the achieved prediction. In outlier explanation, this credit attribution is then used as the feature importance. This approach is, in combination with a classifier, used as a benchmark for the approaches presented by [2] and [1].

The approach by [1], called ATON, determines the feature importance by using a neural network with self-attention that learns to increase the distance between outlier and inlier. They use a triplet-loss by using the outlier to explain and sampling inliers from the neighborhood of that outlier as well as random inliers. After training the network, the self-attention is extracted from the network as the feature importance. They benchmark their approach against other numerical explanation approaches with their approach achieving superior performance.

The approach by [2], called Explainer, is an approach which works similar to isolation forests. It uses isolation trees which are built by performing splits in those leaves that contain the outlier. Leaves are split in such a way that the size of the resulting leaf containing the outlier is as small as possible [2]. At the end of the training, rules are extracted from the individual trees and the rules which are the most frequent among all trees are presented to the user to explain the outliers. The rule frequency is also used to determine the feature importance. This approach claims to work for mixed-type data but is not evaluated for it.

One prevalent way of differentiating outliers from inliers is their distance [21]. This is used in the approach by [1] when determining the local neighborhood of an outlier to select an inlier to differentiate it from. Since their method is designed for numerical data, they utilize simple Euclidean distance. However, the Euclidean distance might not be suitable when determining distances in mixed-type data [21].

Multiple approaches have been suggested to make mixed-type data usable in single-type methods. One-hot encoding is one of the most prevalent methods of allowing mixed-type data to be used in numerical methods [21]. In one-hot encoding, a set of binary dimensions is created for each categorical feature where each dimension stands for one unique feature value within the categorical features.

There are numerous distance metrics which have been designed for numerical data such as Euclidean or Manhattan distance [21]. Only a few have been designed for mixed-type data. One metric specifically designed for mixed-type data is the Gower distance [22].

III. Implementation

To develop an outlier explanation approach for mixed-type data, we used the currently best-performing method on numerical data, called ATON [1], as a foundation. We call the overarching family of our approaches *MIXATON*. We then designed multiple variations of this approach while incorporating specific adjustments for mixed-type data. The proposed methods are *MIXATON_OE_SUM*, *MIXATON_OE_AVG*, *MIXATON_GD* and *MIXATON_EL*.

1. *MIXATON_OE_SUM* This approach works by first encoding the categorical features using one-hot encoding and joining them with the numerical features. The neighborhood search for identifying inlier samples from the neighborhood of the outlier is then performed on this preprocessed dataset using k nearest neighbor with Euclidean distance. These, together with random inlier samples and the outlier are subsequently used to train the network. Afterwards, the learned self-attention is extracted from the network. To obtain the feature importance for the categorical features from the learned self-attention, the attention is added up for each created one-hot encoded feature per categorical feature. For the numerical features, the obtained attention can be used directly as importance. This way, one unique feature importance value is obtained per feature.

2. *MIXATON_OE_AVG* This variant is mostly equivalent to the *MIXATON_OE_SUM* approach. The key difference is that the attention for one-hot encoded categorical features is not summed but instead averaged to obtain the final importance of each categorical feature. This is done to prevent a potential overweighing of categorical features that could occur in the summing approach.

3. *MIXATON_GD* This approach addresses the potential unsuitability of using the Euclidean distance in the neighborhood search on one-hot encoded data for determining samples. This is done by using the Gower distance, as it is a distance metric specifically designed for mixed-type data, on the unprocessed dataset for determining the training samples. Only after determining the samples, the data is one-hot encoded and used for training the network.

4. *MIXATON_EL* The final variant we propose is the *MIXATON_EL* approach. In this approach, the categorical features are one-hot encoded in the beginning before the neighborhood search. To mitigate the effect of the resulting one-hot encoded features being seen as independent [23], embedding layers are used in this approach. The neural network is amended with one embedding layer for each categorical feature. The layer size is chosen for each categorical feature as half its cardinality. After the training of the network, the importance for the categorical features is averaged based on the resulting performance for each embedding layer.

To benchmark our developed variations, we used the only method which has been explicitly proposed for mixed-type data so far, the Explainer approach by [2] as well as XGBoost in combination with SHAP.

IV. Evaluation

In our evaluation, we first created suitable datasets on which we subsequently applied the different approaches.

A. Creating suitable datasets for evaluation

To be able to evaluate the explanation methods, we required both knowledge about which entries within the dataset are outlying as well as which features make these entries outlying. To achieve this, we followed two different approaches.

1) Synthetic outliers

For our first method, we adapt an approach which has been used for creating evaluation datasets in mixed-type outlier detection studies [11, 12]. In this approach datasets without obvious outliers are used and then artificial outliers are injected by shifting values in numerical features and swapping categories in categorical features. This way both the outliers as well as the responsible features are available as a ground-truth. The approach works by randomly selecting 10% of entries in the dataset. We then randomly selected 30% of the features of each of those entries. If the feature is numeric its value gets shifted by two times the feature's standard deviation. In cases in which the feature is categorical or binary the feature value is replaced by another value of that feature. The datasets we used are the Australian credit (a_credit), German credit (g_credit), Heart, Thoracic surgery (thoracic), Auto MPG and Contraceptive (contra) datasets from the UCI ML repository (https://archive.ics.uci.edu/ml/index.php).

2) Pseudo-ground-truth for real outliers

The advantage of using injected outliers is that it is objectively clear what caused the outliers but they might not accurately reflect what real outliers look like. To address this potential limitation, we created a second benchmarking dataset using datasets with real outliers, also from the UCI ML repository.

To generate pseudo-ground-truth explanations, all feature subspaces of these datasets were created and the outliers were scored using IForest, with one-hot encoded categorical features, as well as SPAD [24] and MIXMAD [12] as mixed-type outlier detection methods. Since it is not certain whether the used methods are dimensionally unbiased [16], the rank of the outlier in each subspace is used to determine the explaining subspace. This way, three ground-truth subspaces are obtained for each of the datasets. Since all subspaces had to be created for this approach, we only selected datasets with a dimensionality of 15 or less. This is done because with 16 dimensions or more over 65,000 subspaces would have to be scored using each method which would have been computationally infeasible for multiple datasets.

B. Conducting the evaluation

We evaluated our proposed approaches as well as the already existing method by [2], called *Explainer*, as well as *XGBoost* in combination with *SHAP* on the created datasets. We used multiple metrics to compare the methods' performance. All evaluated explanation methods return an ordered list of explaining features which we used, together with the ground-truth labels, to determine the performance. For providing a fair comparison with the *Explainer* method which only returns a limited subset of features, we employed R-Precision as a measure [25]. Using this metric, the Explainer method did not provide good results. The rigid way of splitting for specific feature values on categorical data might not be able to account for more complex outliers and thus leads to worse performance.

Since the *Explainer* method showcased the lowest performance and it does not return a ranking for the full featurespace, we are comparing the other methods on the complete featurespace using average precision (AP).The result for the synthetic outliers can be seen in TABLE I. When taking the whole feature set into account both the approach of summing the importance and the approach of averaging the importance seem to provide good results on the synthetic outliers.

TABLE I. AVERAGE PRECISION ON SYNTHETIC OUTLIERS

	XGBoost SHAP	MIXATON_ OE_SUM	MIXATON_ GD	MIXATON_ OE_AVG	MIXATON_ EL
a_credit	0.3928	0.4091	0.4153	**0.4193**	0.4047
autompg	0.4457	**0.4845**	0.4276	0.4468	0.4591
contra	0.5283	0.5191	0.4954	**0.5575**	0.4002
g_credit	0.4090	**0.4250**	0.4202	0.4237	0.4165
heart	0.3551	**0.4035**	0.3521	0.4022	0.4023
thoracic	0.3775	**0.4182**	0.4145	0.4098	0.3945
avg	0.4181	**0.4432**	0.4208	**0.4432**	0.4129

The result rated via the AP for the real outliers can be seen in TABLE II. All *MIXATON* approaches provide good performance with the method using the Gower distance achieved the best performance.

TABLE II. AVERAGE PRECISION ON REAL OUTLIERS

Dataset	XGBoost SHAP	MIXATON_ OE_SUM	MIXATON_ GD	MIXATON_ OE_AVG	MIXATON_ EL
abalone_iforest	0.4468	0.4218	0.4162	0.4309	**0.5271**
abalone_mixmad	0.5308	0.6089	0.5973	0.6308	**0.6382**
abalone_spad	0.3486	0.4694	0.4747	0.4577	**0.4996**
adap_iforest	**0.4091**	0.3486	0.3766	0.4109	0.3721
adap_mixmad	0.4067	**0.4320**	0.4285	0.3569	0.3961
adap_spad	**0.3465**	0.2734	0.3225	0.3400	0.2931
credit_iforest	0.4112	**0.4440**	0.4297	0.3894	0.3871
credit_mixmad	**0.4522**	0.4492	0.4333	0.4094	0.4115
credit_spad	0.3438	0.3863	**0.3858**	0.3526	0.3460
heart_iforest	0.3642	0.3982	**0.4096**	0.3881	0.3937
heart_mixmad	0.4445	0.4498	**0.4693**	0.4526	0.4616
heart_spad	0.3680	0.3923	**0.4069**	0.3529	0.3628
mammography_iforest	0.6389	0.6870	**0.7124**	0.6327	0.5444
mammography_mixmad	0.6995	0.7406	0.7254	**0.7426**	0.6492
mammography_spad	0.6483	0.6705	**0.6786**	0.6554	0.5889
avg	0.4573	0.4781	**0.4844**	0.4669	0.4581

Some of the key observations for the overall results are:

1. No single approach is always superior or inferior. This is to be expected since each data set has different characteristics.
2. Certain approaches provide consistently better performance on a large variety of ground truth labels and datasets. The *MIXATON_GD*, *MIXATON_OE_SUM* and

MIXATON_OE_AVG approaches provide the best performance. This is notable, since these approaches have been introduced for the first time in this paper.

3. There are some approaches that are stronger on certain datasets while other approaches are stronger on other datasets (XGBoost and SHAP, *MIXATON_GD*). This insight means that it might be possible to construct approaches that utilize the strength of multiple approaches to obtain a superior performance.
4. The result of methods might depend on how important either the numerical or categorical features are for making a certain entry an outlier. Thus, the method *MIXATON_OE_SUM* might perform better on a dataset in which the categorical features are more important for making an outlier outlying while *MIXATON_OE_AVG* might perform better when the responsibility is evenly distributed between the feature types. This should be further investigated in future research.

V. Conclusion

In this paper, we propose multiple approaches for explaining outliers on mixed-type data. We also perform the first benchmark of outlier explanation approaches on mixed-type data and describe two approaches for preparing suitable benchmark datasets for this purpose. We show that our proposed approaches *MIXATON_OE_SUM*, *MIXATON_OE_AVG* and *MIXATON_GD* show good performance on various datasets and outperform previously introduced approaches. Using these methods, outlier detection can be made more useful in domains in which it is applied on mixed-type data.

One important question that is raised in this comparison of multiple approaches is: *Which explanation method is the most suitable for explaining my detected outliers?* Our experimental findings suggest that there is no one best performing outlier explanation measure for all mixed-type datasets. Although, the different *MIXATON* variations we propose, apart from *MIXATON_EL*, all seem to provide consistently good performance on most datasets.

One possible approach to mitigate the fact that different methods perform better on different datasets would be to combine explanation methods in an ensemble approach. This should be investigated in future research.

References

[1] H. Xu *et al.*, "Beyond Outlier Detection: Outlier Interpretation by Attention-Guided Triplet Deviation Network," in *Proceedings of the Web Conference 2021*, 2021, pp. 1328–1339. [Online]. Available: https://doi.org/10.1145/3442381.3449868

[2] M. Kopp, T. Pevný, and M. Holeňa, "Anomaly explanation with random forests," *Expert Systems with Applications*, vol. 149, pp. 1–18, 2020, doi: 10.1016/j.eswa.2020.113187.

[3] D. Samariya, S. Aryal, K. M. Ting, and J. Ma, "A New Effective and Efficient Measure for Outlying Aspect Mining," in *Web Information Systems Engineering – WISE 2020*, 2020, pp. 463–474.

[4] M. A. Siddiqui, A. Fern, T. G. Dietterich, and W.-K. Wong, "Sequential Feature Explanations for Anomaly Detection," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 1, Article 1, 2019, doi: 10.1145/3230666.

[5] N. Liu, D. Shin, and X. Hu, "Contextual outlier interpretation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden: AAAI Press, 2018, pp. 2461–2467.

[6] M. Carletti, M. Terzi, and G. A. Susto, *Interpretable Anomaly Detection with DIFFI: Depth-based Feature Importance for the Isolation Forest*.

[7] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation Forest," 2008, doi: 10.1109/ICDM.2008.17.

[8] J. Nonnenmacher, F. Kruse, G. Schumann, and J. Marx Gómez, "Using Autoencoders for Data-Driven Analysis in Internal Auditing," *HICSS*, 2021, doi: 10.24251/HICSS.2021.697.

[9] H. Xia, H. Q. Vu, J. Tan, X. Li, and G. Li, "Characterizing the Outlying Feature Set of Groups," *Procedia Computer Science*, vol. 165, pp. 119–125, 2019, doi: 10.1016/j.procs.2020.01.086.

[10] F. Angiulli, F. Fassetti, and L. Palopoli, "Detecting Outlying Properties of Exceptional Objects," *ACM Trans. Database Syst.*, vol. 34, no. 1, 2009, doi: 10.1145/1508857.1508864.

[11] S. Eduardo, A. Nazabal, C. K. I. Williams, and C. Sutton, "Robust Variational Autoencoders for Outlier Detection and Repair of Mixed-Type Data," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, S. Chiappa and R. Calandra, Eds., Proceedings of Machine Learning Research: PMLR, 2020, 4056--4066. [Online]. Available: http://proceedings.mlr.press/

[12] K. Do, T. Tran, and S. Venkatesh, "Energy-based anomaly detection for mixed data," *Knowledge and Information Systems*, vol. 57, no. 2, pp. 413–435, 2018, doi: 10.1007/s10115-018-1168-z.

[13] M. Garchery and M. Granitzer, "On the influence of categorical features in ranking anomalies using mixed data," *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia*, vol. 126, pp. 77–86, 2018, doi: 10.1016/j.procs.2018.07.211.

[14] J. Zhang and H. Wang, "Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance," *Knowledge and Information Systems*, vol. 10, no. 3, pp. 333–355, 2006, doi: 10.1007/s10115-006-0020-z.

[15] N. X. Vinh *et al.*, "Discovering outlying aspects in large datasets," *Data Mining and Knowledge Discovery*, vol. 30, no. 6, pp. 1520–1555, 2016, doi: 10.1007/s10618-016-0453-2.

[16] L. Duan, G. Tang, J. Pei, J. Bailey, A. Campbell, and C. Tang, "Mining outlying aspects on numeric data," *Data Mining and Knowledge Discovery*, vol. 29, no. 5, pp. 1116–1151, 2015, doi: 10.1007/s10618-014-0398-2.

[17] B. Micenková, R. T. Ng, X. Dang, and I. Assent, "Explaining Outliers by Subspace Separability," 2013, doi: 10.1109/ICDM.2013.132.

[18] X. H. Dang, B. Micenková, I. Assent, and R. T. Ng, "Local Outlier Detection with Interpretation," in *Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 304–320.

[19] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert, "Discriminative features for identifying and interpreting outliers," *2014 IEEE 30th International Conference on Data Engineering*, 2014.

[20] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[21] C. C. Aggarwal, *Outlier analysis*: Springer, 2017.

[22] T. Wangchamhan, S. Chiewchanwattana, and K. Sunat, "Efficient algorithms based on the k-means and Chaotic League Championship Algorithm for numeric, categorical, and mixed-type data clustering," *Expert Systems with Applications*, vol. 90, pp. 146–167, 2017, doi: 10.1016/j.eswa.2017.08.004.

[23] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2020, doi: 10.1109/TKDE.2020.2992529.

[24] S. Aryal, K. M. Ting, and G. Haffari, "Revisiting Attribute Independence Assumption in Probabilistic Unsupervised Anomaly Detection," in *Intelligence and Security Informatics*, 2016, pp. 73–86.

[25] G. O. Campos *et al.*, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016, doi: 10.1007/s10618-015-0444-8.

Log Sinks and Big Data Analytics along with User Experience Monitoring to Tell a Fuller Story

Vidroha Debroy, Senecca Miller, Mark Blake, Alex Hibbard and Cody Beavers
Product and Development
Dottid
Dallas, Texas, USA
{vidroha.debroy, senecca.miller, mark.blake, alex.hibbard, cody.beavers}@dottid.com

***Abstract*—Understanding and continually improving the user experience is critical to the success of web applications, especially those that are business-driven. There exist a multitude of tools to monitor user activities on a website, which then provide metrics to help developers and company leadership understand where their users experience pain-points. However, all tools suffer from their own limitations and ultimately it is important for companies to have as much control (as possible) over all data collected by third-party tools, and be able to make decisions related to its storage, retention, processing, etc., in an agile manner. At *Dottid*, we use a third-party tool to understand and enhance the user experience on our web application, recently however, we ran into a problem regarding failed logins that required us to architect a solution ourselves, since the tool cannot address this issue for us. The solution leverages log collection and big data analytics and its architecture has paved the way for us to build more actionable insights than if we were using the tool as-is. We transparently share our experiences, and the details of our solution and rationale, with the goal of benefiting others, and promoting further industry-academic collaboration, in this space.**

Keywords: Software, Logging, Monitoring, Big Data, Analytics

I. INTRODUCTION

Creating a good User Interface (UI) and User Experience (UX) is very important, especially given that more and more products and services are now sold over the internet [12]. The focus on Human-Computer Interaction (HCI) and Interaction Design (IxD) – fields acknowledging that understanding users is crucial to successful design for all interactive products" – is also not new to academia [11]. Indeed, from a user experience standpoint, a lot of care needs to be taken when designing web apps[1] as opposed to web sites (the former typically offering a wide range of interactive features and dynamic information and the latter offering mostly static content).

Dottid [1] is a tech company in the business domain of Commercial Real Estate (CRE) and we are relatively small-sized (less than 50 employees) and relatively new (less than 5 years in business). Our core product is a web application that follows a microservice architecture, and we run in the cloud (Google Cloud Platform, i.e., GCP) to leverage benefits such as economies of scale. While a lot of business in commercial real estate has historically taken place over in-person meetings and over physical media such as paper; our company revolutionizes the model by providing an online platform to organize deals, expedite transactions and accelerate time to revenue. Understandably ensuring our users have a good experience when navigating our application is very important to us and thus, we leverage real-time user monitoring[2] wherever possible.

This is achieved by using the out-of-the-box capabilities provided by GCP, but we also rely on a third-party tool named *fullstory* [7]. By integrating some code into our application, we stream information on user interactions back to *fullstory* where we can view all the data collected in a consolidated way (this includes security and privacy controls over what data is collected and who can view it). One of the more exciting features of *fullstory* is the record-and-replay functionality (also known as session replay) which allows us to reproduce events such as mouse movements, clicks, scrolling, etc., exactly as the user performed them and better understand the resultant behavior of our app exactly as the user experienced it.

It should be noted though that user experience monitoring is most meaningful in the context of an actual user, i.e., we care the most about our actual customers. For this reason, we only send data to *fullstory* after a valid user has authenticated (i.e., successful login) and is inside our web app. This leads to an interesting question – *what about the user experience when trying to log into our web app*? Addressing this was a real issue that we had to deal with as it was directly affecting some of our customers. There is more to this then simply letting the user know if their login was unsuccessful or offering them a way to reset their password. There are ramifications in terms of how to track the data, store the history, etc., and more interestingly how to connect it to activities after the user successfully logs in and then derive further actionable information. This paper serves to discuss the solution that we came up with, and currently employ, at Dottid and discuss our rationale. In doing so, we hope to help others that might be in a similar situation as well as provide industry-experience in an academic publication.

[1] In the interests of brevity, we used 'application' and 'app' interchangeably in this paper, both in the singular and plural sense.

[2] In a lot of discussions 'monitoring' implies performance monitoring of an app, which is different from understanding how users interact with an app (our focus). *Application Performance Monitoring* (APM) [6] is certainly correlated with User Experience, but we clarify that this is not the focus of our paper.

DOI 10.18293/SEKE2022-175

II. BACKGROUND AND MOTIVATION

A. *Web Application Setup*

Our web app is microservice-based and is hosted in the cloud along with dependencies. Our cloud of choice is Google Cloud Platform (GCP) – however, all architectural choices are cloud agnostic. Stated differently, we will not need to re-design our application for a different cloud provider, and this also means the discussions in this paper are not limited to CGP and can be generalized irrespective of cloud provider.

B. *Integration with fullstory*

We follow the standard model for integrating with *fullstory* in that our client-side code contains a snippet of JavaScript code provided by *fullstory* and once this script code loads it captures all web-based interactions and mutations. We only specify what data we want to collect and what to ignore, and we leverage features such as masking, and apply protections as to who can access what data. These are features that are built-into *fullstory* and offered to all of their clients. All data is stored on the *fullstory* side, and we access the data using their web application which is also where we view session recordings and leverage the dashboards and analytics that *fullstory* provides.

C. *Understanding the Problem*

Much like other web apps we have a standard login screen that is shown in Figure 1. We note that at this point, i.e., the login screen, *fullstory* is not collecting data (rather data is not being collected for *fullstory*). Also, for all practical intents and purposes, anyone on this screen is not a recognized user (we have not yet identified them as such). Which then leads to the problem – if someone runs into an issue on this page/screen, what would they do and how is the data we collect useful?

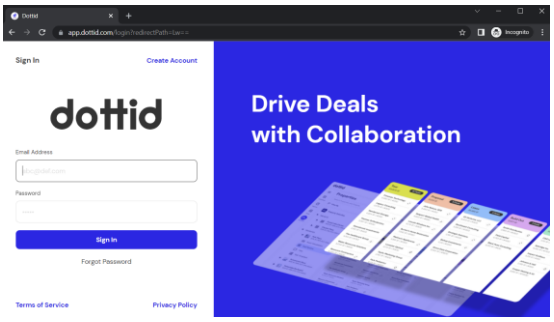


Figure 1. The standard login screen to our web app

Identifying the actual cause is not the problem and neither is this the focus of our paper. The real problem is how this hampers the user experience, and the question is beyond just identifying the cause of the issue, but rather how to convey it[3] and then come up with requisite actions to solve the problem.

The problem is only half-solved if this is a continually repeated exercise every time it occurs, and this stops us from being an Agile development team, or an Agile company for that matter. Also, if our customer representative needs to always contact an administrator on our side to look for further information, then it means a delay in resolution and also more frustration for our customer/user, which is definitely something we want to avoid. Unfortunately, as discussed next, we faced this same exact situation and *fullstory* could not be used to solve this problem, and neither could the out-of-the-box cloud (GCP) monitoring in a way that was acceptable to us.

We rely on *Cloud Identity* and related-technologies offered by GCP to track our authentication information and it is true that audit logs will confirm when an attempt was made to login and upon failure, why that failure occurred. But at the same time, a limitation is that one needs to be an administrator in order to access the audit logs [14]. So, with the existing setup we ran into a very-real issue: *if a user had trouble logging in, and they felt that they were inputting the correct credentials, how would the customer representative provably identify the issue and the fix*? Granting all of our customer representatives with admin-level access was infeasible; at the same time, having our customer representatives always contact our administrators to resolve login issues was too slow (both in action and in response) and did not make for a good customer experience. We needed a solution for this.

III. CONSTRUCTING A SOLUTION

A. *Logging User Activities*

The first part of our solution stems from the intuition that when leveraging the cloud for authentication, all such (login) activities are logged, if not for our purposes, then for auditing and debugging purposes on the cloud provider's side. If we were to transfer those logs, then we could have finer-grained control over the data within. So, the very first thing we did is to enable that level of logging across the Identity Platform for each our environments of interest in GCP. On the surface this may look like just checking a checkbox on an interface (as shown in Figure 2), but it comes with significant consequences.

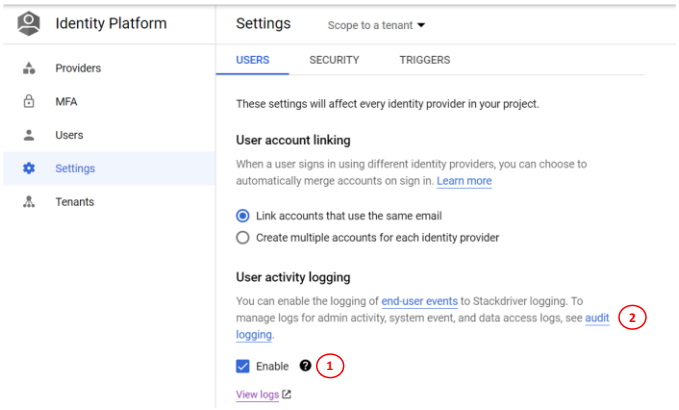


Figure 2. User activity logging in GCP

First, with respect to the figure, checking that box (annotated as 1) does not address everything as is described by the link to 'audit logging' (annotated as 2) and some non-trivial configuration is still needed. Second, the checkbox is representative in intent but not in where the data goes in terms

[3] The conveyance of this information is very important in the interests of security and avoiding attacks: some companies may want to indicate that a user does not exist when a bad username is entered, and some companies may not want to divulge that. Similarly, some companies may want to indicate a bad password, some companies may want to provide a generic error message.

of identifying a destination for consuming that information. Furthermore, since the information exists just in unprocessed logs – we incur charges not just in terms of storage, but also in terms of queries run against said logs; especially relevant if we were looking for a particular user (since it would be a pure text-based search). Thus, this gives us the data we want but not in a format we can consume in a cost-effective manner.

B. *Converting to the Consumable*

The next part of our solution focuses on directing these logs and then transforming them into *queryable* data. Since the logs are officially owned by us, we can pipe these to storage (that we own) and can establish filters on what is piped in and what is not (to reduce the data footprint). The first thing we did is to figure out where to store the data and we went with *BigQuery* [2]. BigQuery represents a highly scalable, performant and flexible data warehouse solution and provides an *Application Programmer Interface* (API) for interacting with one's data and represents a full-scale data storage and analytics solution. ***Sinks*** [4] control how logs are routed to all their supposed destinations. We defined a sink with inclusion rules to focus only on Identity Logs and piped them into a *BigQuery* dataset. This represented a truly composite and scalable solution, much more so because the transformation in this manner *allows us to query the text-based logs in Structured Query Language (SQL)* which is a very common and popular language/standard.

TABLE I. SIMPLIFIED VERSION OF OUR LOG SINK

```
gcloud logging sinks describe masked-login-sink
bigqueryOptions:
  usePartitionedTables: true
  usesTimestampColumnPartitioning: true
description: Piping login data from logs into BigQuery.
destination:
bigquery.googleapis.com/projects/masked/datasets/ masked_ds
filter: logName=
"projects/masked/logs/identitytoolkit.googleapis.com/requests"
writerIdentity:
serviceAccount:masked@gcp-sa-logging.iam.gserviceaccount.com
```

Sinks are true cloud resources, with definitions that can be maintained within source control, which make them especially attractive. To illustrate this and the earlier feature descriptions, TABLE I lists a simplified definition of our log sink with details regarding project/dataset or account names intentionally obfuscated (using the word '*masked*') for privacy reasons.

C. *Maximizing the Benefits of Data Analytics*

Beyond just making the data easy to query - by piping our logs in real-time to alternative storage, we maintain a simple rolling-history on the text-based logs themselves which results in cost-savings since the log-sizes always stay small (we only retain a minimal history). Furthermore, by creating partitioned tables we can divide our data into segments that are easier to manage and query, for example: based on time periods. This in turn reduces the size of individual queries (in terms of the number of bytes) which consequently reduces costs. The prior discussions are best appreciated visually and by using a real example and so in TABLE II we present the actual query to find login information on a user via their email address (one of the authors simulated a failed login by intentionally supplying a valid username but a bad password). The query is written as standard SQL and can run verbatim against our production environment with only one exception: for privacy reasons, as before, we again obfuscate the name of our actual GCP project name and dataset name, using the term '*masked*' instead.

TABLE II. QUERYING FOR LOGIN INFORMATION BY EMAIL

```
select * from (select COALESCE(
jsonpayload_logging_requestlog.request.email,
jsonpayload_logging_requestlog.metadata.tokeninfo.claims.email,
jsonpayload_logging_requestlog.response.email) as `UserEmail`,
severity,
timestamp,
jsonpayload_logging_requestlog.methodname as `MethodName`,
jsonpayload_logging_requestlog.status.message
from `masked.masked_ds.identitytoolkit_googleapis_com_requests`
WHERE DATE(timestamp) >= "2022-03-24"
order by timestamp desc)
where UserEmail = 'vidroha.debroy@dottid.com'
```

This query does quite a bit – it coalesces 3 different sources of email info; looks up severity, timestamp and any message information along with the name of the GCP Identity method; and it filters this down to on or after March the 24th, for just the email searched (*vidroha.debroy@dottid.com*); finally ordering any data in descending order of timestamp. In the interests of brevity, only 2 sample rows of output are shown in Figure 3, but it can be clearly seen from the 2nd row that there was a failed login for this user due to an incorrectly supplied password, while the 1st row shows normal interaction.

Figure 3. Results of the query

Even more exciting - thanks to the power of *BigQuery*, we receive a lot of useful meta-data on the query and its execution that can be used for subsequent optimization; we can save the query and trigger alerts based off of the results; we can export the results to different formats, and automatically build charting using the results as shown in Figure 4.

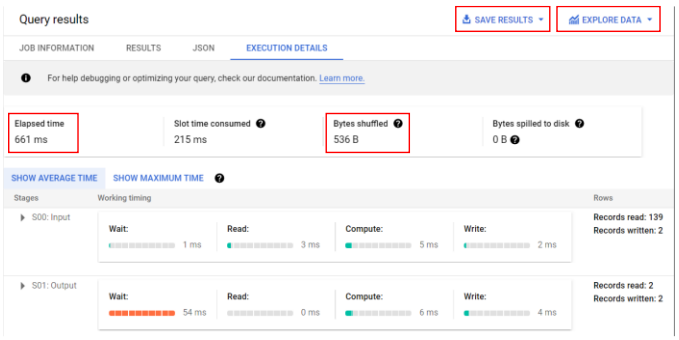


Figure 4. Meta-data on query execution and other options

D. Telling a Fuller Story

As discussed earlier, *fullstory* offers us useful ways to track user activity once they have logged into our app. At the time of writing of this paper, we are actively researching approaches to export data out of *fullstory* [10] and store it in a cost-effective way. The approach presented in this paper helps us solve user issues when they haven't yet logged into our app (which cannot be tracked by *fullstory*). In this manner, it may seem like these are opposite sides of the same coin (tracking before login and tracking after login), and in many ways this is true. In fact, for all practical purposes they represent completely independent datasets, and the only real piece of information that connects a user from one dataset to the other is the username[4] (which is an email address in our case and thus, is guaranteeably unique).

But by adopting big data analytics we can now establish meaningful correlations where it was otherwise very difficult to do so. Both datasets track timestamps, IP addresses, user agent strings etc. which can be used to match up related events even in the absence of referential integrity. For example, if a user enters a bad username at log in – by looking at the source IP address, which Operating System/Browser was used, etc., we can infer (with some degree of confidence) whether these were from a valid user or some malicious attack. And for valid users we can examine whether it is manual error or ask ourselves why users are having trouble logging in (maybe users click in the wrong spot due to a layout problem) and then identify improvements to our own user-interface to make it more usable and friendly. Thus, using big data analytics allows us to learn more about the end-to-end experiences of our users, thereby, telling a much fuller story.

IV. RELATED WORK

Usability is a critical aspect in interactive software systems [16] and has been recognized as an important factor in the acceptance of software by end users [5]. However, even with decades of research, there is still a debate about the relationship between usability and user experience [8]. A/B Testing is a technique employed in practice to evaluate partial functionality as well as how users respond to new features. Taking data into account is a big aspect of such endeavors [13] and they help software developers understand their users, much as we rely on *fullstory*. Our focus is on very specific goals: improving the experience for users who are unable to login, as well as tying that to disparate data sets for users who have logged in.

Understanding and processing the data from a usability testing perspective is also an important concern and research has been conducted on how to make sense of the data [9]. While similar in intent, our work is contrasted in that our focus is on how to leverage data collected from real users in our production environment. The idea of using big data analytics for user activity analysis is relatively new [15] and it has been noted that applying processing techniques such as machine learning to UX research has received little academic attention [3]. We share the intent to draw attention, further the literature, and promote industry-academic collaboration in this space.

[4] This does not imply that there is any key-based referential integrity in any tables from the login info dataset to the *fullstory* dataset based on the username, it only suggests that this piece of data exists in both datasets.

V. CONCLUSION

We discuss our approach at *Dottid* to address a problem that real users were running into related to failed logins. While we utilize a third-party user-monitoring tool, it proved to be ineffective in terms of solving the problem at hand. Since we run in the cloud (GCP), we leveraged cloud-oriented solutions such as logging (log sinks) and big data analytics (*BigQuery*) to bridge the gap between what used to be two independent (yet related) sides of the story – users who had logged in and users who might experience trouble logging in. We transparently share details of our approach to help others in similar situations and promote industry-academic collaboration. Future work includes applying machine learning and related techniques, to analyze the volume of data we collect, for actionable insights.

REFERENCES

[1] About – Dottid. https://dottid.com/about-dottid, last accessed March 14th 2022.

[2] BigQuery – GCP. https://cloud.google.com/bigquery, last accessed March 15th 2022.

[3] M. Chromik, F. Lachner and A. Butz, "ML for UX? – an inventory and predictions on the ser of machine learning techniques for UX research", in *Proc. of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society* (NORDCHI), Tallinn, Estonia, October 2020.

[4] Configure and Manage Sinks – GCP. https://cloud.google.com/logging/docs/export/configure_export_v2, last accessed March 18th 2022.

[5] L.M. Cysneiros and A. Kushniruk, "Bringing usability to the early stages of software development", in *Proc. of the 11th IEEE Intl. Requirements Engineering Conference*, Monterey Bay CA, USA, September 2003.

[6] V. Debroy, A. Mansoori, J. Haleblian and M. Wilkens, "Challenges faced with application performance monitoring (APM) when migrating to the cloud", in *Proc. of the IEEE International Symposium on Software Reliability Engineering Workshops* (*ISSREW*), pp. 153-154, Portugal, October 2020.

[7] Digital Experience Intelligence – fullstory. https://www.fullstory.com/digital-experience-intelligence-platform/, last accessed March 15th 2022.

[8] T. Haaksma, M de Jong, and J. Karreman, "Users' Personal Conceptions of Usability and User Experience of Electronic and Software Products", *IEEE Software*, pp. 116-132 vol. 61, issue 2, June 2018.

[9] W. Haiyan and Y. Baozhu, "A data-processing mechanism for scenario-based usability testing", in *Proc. of the 2nd IEEE Intl. Conference on Computing, Control and Industrial Engineering*, China, August 2011.

[10] Fullstory – segment exports. https://developer.fullstory.com/create-segment-export, last accessed March 28th 2022.

[11] A. Granic, "Technology in use: the importance of good interface design", in *Proc. of the Intl. Conference on Infocom Technologies and Unmanned Systems* (ICTUS), pp 43-49, Dubai, UAE, December 2017.

[12] R. Gunawan, G. Anthony and M. Vendly, "The effect of design user interface (UI) e-commerce on user experience (UX)", in *Proc. of the 6th Intl. Conference on New Media Studies* (CONMEDIA), pp. 95-98, Tangerang, Indonesia, October, 2021.

[13] R. King, E. Churchill and C. Tan, "Designing with data", O'Reilly Media Inc, April 2017.

[14] Login audit – Help. https://support.google.com/a/answer/4580120, last accessed March 15th 2022.

[15] M. Parwez, D. Rawat and M. Garuba, "Big data analytics for user-activity analysis and user-anamoly detection in mobile wireless networks", in *IEEE Transactons on Industrial Informations,* 13(4): 2058-2065, January 2017.

[16] R. Ren, J.W. Castro, S. Acuna and J. Lara, "Usability of chatbots: a systematic mapping study", in *Proc. of the 31st Intl. Conference on Software Engineering and Knowledge Engineering* (SEKE), Lisbon, Portugal, July 2019.

A Node-Merging based Approach for Generating iStar Models from User Stories

Chao Wu[1], Chunhui Wang*[1], Tong Li[2] and Ye Zhai[1]

[1]College of Computer Science and Technology,Inner Mongolia Normal University,Hohhot, China
wuchao@mails.imnu.edu.cn, ciecwch@imnu.edu.cn, cieczy@imnu.edu.cn
[2]Faculty of Information Technology,Beijing University of Technology, Bejing, China
litong@bjut.edu.cn

***Abstract*—User story is a widely adopted requirement notation in agile development. Generally, user stories are written by customers or users in natural language with limited form to describe user's needs for the software system from their perspectives. However, since user stories are generally presented in a flat list, the relations derived from the user stories are difficult to capture. It reduces the understanding of the system as a whole. One solution to this problem is to build goal-oriented models that provide explicit relations among user stories. But extracting concepts and relationships from a large number of discrete user stories often take a lot of time for the agile development team. This paper proposes an iStar model generating approach based on node-merging from user stories. The method first extracts the iStar nodes from the semi-structured user stories, then uses a BERT (Bidirectional Encoder Representations from Transformers) model to measure the similarity between the nodes, and then nodes to be merged are identified and the edges between the iStar nodes are connected. Experiments are designed to illustrate the effectiveness of the proposed approach.**

***Index Terms*—User story, iStar, Model construction**

I. Introduction

Agile software development includes a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams [1]. It advocates adaptive planning, evolutionary development and delivery, and encourages rapid and flexible response to changes [2]. These features have led to agile methods to achieve remarkable success in software industry.

User story is a widely adopted requirement notation in agile development. Generally, user stories are written by customers or users in natural language with limited format that illustrates requirements from the user's perspective. A general user story pattern relates a **who**, a **what** and possibly a **why** dimension, and uses keywords arranged these dimensions into one sentence (e.g., Cohn suggests a user story pattern [3]: *As a < type of user>, I want <some intention>, [so that <some reason>]*). Although a user story is short and simple descriptions, it normally consist of the following elements: *Role* (the aspect of who representation), *Goal/task* (a circumstance described by roles or specific things that must be done), and *quality* (expectations of the customer for the quality of the final product).

One challenge that is not addressed by user stories is to capture the relations of the user stories. For example, it is difficult to know the relations of decomposition and hierarchy between user stories. Such relations help the developers to better understand and structure the backlog items between user stories.

Goal-oriented modeling allows for the clear and explicit dependencies of goals to facilitate understanding of stakeholder needs, dependencies, etc. Some studies have observed that some concepts and relationships in user stories can potentially be aligned with goal models, and proposed the transformation approach from user stories to goal models [4]–[6].

Building goal models from user stories takes a lot of time and human overhead when user stories are large in scale. An automated goal-oriented model extraction approach is proposed in [7]. They propose 3 heuristic rules to identify refinement relationships between the **what** parts in user stories by using natural language processing (NLP) techniques. Their approach ignores the information in the **why** part of the user story and also ignores the relationship between the **what** and **why** parts. And this information is necessary to identify the model concepts and the hidden hierarchical relationships.

iStar model is a goal model that has been applied in many areas, e.g., healthcare, security analysis, eCommerce [8]. It focuses on the intentional (**why**), social (**who**), and strategic (**how**) dimensions. The concepts of iStar model correspond to the concepts of user story, and can express the composition and hierarchical relationships between user story concepts.

In [9], we proposed a preliminary framework for constructing iStar models from user stories and summarized 5 heuristic rules for extracting iStar elements from input user stories. In this paper, we focuse on the refinement relationship between iStar nodes. The key to identifying the refinement relationships is a node-merging based approach. This approach pays attention to the nodes with similar description in the model, and combines them based on merging rules. The process of this approach consists of three steps: model node identification, node merging and edge identification. Model node identification extracts iStar nodes from a set of user stories by using NLP techniques. Node merging combines

DOI 10.18293/SEKE2022-176
*Corresponding Author

the similar nodes by using a node similarity measurement approach based on BERT mode. Edge identification can find the edges between the iStar nodes by using three rules and finally obtains an initial iStar model. Several experiments are designed to evaluate our approach.

The rest of the paper is organized as follows. Section 2 outlines the relevant work. Section 3 details the process of building an initial iStar model from user stories. Section 4 performs this process with a small example. Section 5 evaluates the effectiveness of our proposed approach by using three real user story datasets from different domains. Section 6 summarizes the paper.

II. Related work

A. From user strories to goal models

There has been some research focusing on the transformation from user stories to goal models. However, since existing goal oriented modeling methods tend to be overly complex for non-technical stakeholders, some researchers propose some simplified goal models. Wautelet et al. [4] proposed a rationale diagram to build various trees of relating user story elements in a single project. Lin et al. [5] use a light weight goal net to model user stories. The goal net supports goal selection and action selection mechanisms and provides flexibility to task selection and process optimization in agile software development.

To build goal models from user stories, some studies define heuristic rules for the transition from user story templates to model elements. Jaqueira et al. [10] propose role in user story is mapping to actor in istar model, action in user story is mapping to task in goal model, and goal is mapping to goal in goal model. Gune et al. [7] propose to automatically generate a goal model from a set of user stories by applying NLP techniques and 3 heuristics. The heuristics focuses on grouping the user stories around common objects or verbs.

Our approach also focuses on automated generation from user stories to iStar models. We propose to use node merging to extract refinement relationships between user stories. The key is to compare concepts that can be combined in the iStar model. In addition, we also focus on the identification of iStar nodes and their connected edges. Our goal is to generate a realistic model.

B. Similarity calculation of requirements

Similarity calculation is a key technique for identifying similar concepts and grouping similar user stories. At present, in the field of requirements engineering, there are mainly two types of methods: lexical-based methods and semantic-based methods [11].

Lexical-based methods regard the requirements as a sequence of words and concern with the similar of characters in word and their sequence of characters. There are many algorithms, such as Levenshtein distance [12], Jaccard [13], Hamming distance. Mihany et. al. [11] proposed an automated system for measuring similarity between software requirements to identify reused components. This system uses Dice, Jaccard, and cosine similarity methods to measure the similarity between requirements in order to identify reusable components.

Semantic-based methods consider the meaning of words rather than characters of words. In order to compare the meaning of two words, corpus and some kind of language model are usually used. For example, some researchers use neural network models (e.g. Word2Vec, CNN, BERT) to generate word vector. Arau et al. [14] pre-trained BERT model to generate semantic textual representations to automatically identify software requirements from APP review. They trained seven different applications, and then an eighth application was used to evaluate the model. The results of the evaluation showed that the F1 scores were more than 40%.

In our work, we use the BERT model to generate word embedding for user stories. In order to accommodate the similarity comparison of short texts, we add the corpus of similarity pairs when pre-training the BERT model.

III. A Node-merging Based Approach

In this section, we introduce the proposed approach to generating iStar models from user stories, the process of this method consists of three steps (shown in Fig. 1): iStar node identification, node merging, and edge identification.

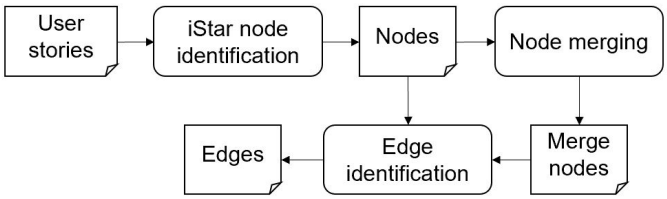


Fig. 1. The process of a node-merging based approach

A. Node identification

In this phase, we identify iStar nodes and their types from a set of input user stories. There are three types of nodes: *role* type, *intention* type, and *quality* type. The detailed definitions of these three types can be found in [8], where the intention nodes includes *goal* nodes and *task*. Since it is difficult for automated methods to distinguish these two concepts, we do not make a distinction between these two concepts in this paper.

Given a set of user stories, we first extract user story components to construct iStar nodes. Each user story in the set has a fixed format. We use the following common user story pattern [3]: *As a < type of user>, I want <some intention>, [So that <some reason>]*. From this pattern, we use NLP techniques to extract their components: **who**, **what** and **why**. These components correspond to the part after the three fields "As a", "I want to" and "so that", respectively.

Specifically, we perform the following NLP steps: (1) Convert case: all nodes were converted in to lowercase. (2) Tokenization: nodes were split into a set of words by

removing spaces and commas. (3) Remove special characters: special characters are usually non-alphabetic and numbered characters. To remove noisy data, we remove special characters using the re regular expressions. (4) Remove stop words: stop words are removed like “a”, “to” etc and remove the keywords “As a” , “ I want” and “ So that”. And in the final step (5) Word Lemmatization: removes the affixes of words depending on part-of-speech (POS) tagging identifies the lexical properties and maps them to their root form to find the original form.

After applying the above steps to a set of user stories, we obtain the user story components, and apply the following 3 rules to extract *role* and *intention* nodes from user stories.

- Each **who** component maps to *role* type node;
- Each **what** component maps to *intention* type node;
- Each **why** component maps to *intention* type node.

In order to discover *quality* type nodes, we query the *quality word* in *intention* type nodes. To obtain the *quality words*, we collected 241 words from 4 related works [15]–[18] that contained a keyword list of non-functional requirements. The adjectives and adverbs were selected among these words and 58 words were obtained, such as easy, high, clear, friendly, early, ugly, fast, and so on.

B. Node merging

Node merging mainly includes three steps: representing node with BERT model, calculation of node similarity, and return of node pairs with high similarity.

We pre-train the BERT model [19] with STSbenchmark dataset of similarity scores (8628 sentence pairs) [20], and its parameters are set to that batch size is 16, epoch is 20, step is 1000, eventually saved fine-tune BERT model for generating the corresponding node vector. Equation 1 is used to measure the similarity between two vectors obtained by BERT model. The similarity result is a value between 0 and 1. For pairs with a similarity higher than 0.8 (see the section V-D for the details of threshold selection), we consider nodes that should be merged and merge them.

$$\cos(x, y) = \frac{x \cdot y}{|x| \cdot |y|} \tag{1}$$

C. Edge identification

In this phase, we automatically identify the edges between nodes based on the identified iStar nodes and the merged nodes. There are three types of edges: *means-ends*, *refinement* and *contribution*. The *means-ends* connects two *intentions* nodes in a user story. The source is from **what** part in the user story. The target is from **why** part in the user story. The detailed definitions of *refinement* type and *contribution* type in this paper show in [8].

3 rules are used to identify the above three types of edges.

- **Rule 1**: Add an edge of type *means-ends* for two intention nodes from the same user story. The source is from **what** part in the user story. The target is from **why** part in the user story.
- **Rule 2**: Add edges of type *refinement* type for a merged node and its components (nodes).
- **Rule 3**: Add an edge of type *contribution* for a *quality* node and its corresponding *intention* node.

IV. Illustrative example:Online Shopping

In this section, we use a case of Book Factory [21] to illustrate the proposed method. The Book Factory is a small online book purchasing system. In the online shopping system, users should be allowed to browse online book details, add books to shopping cart, complete online orders and query logistics information. On the other hand, for the system, it should calculate the order prices before customers pay for the orders. In addition, online payments are processed through the Ogone payment platform to improve payment security. For this group of user stories, we unified the user story template (using the key words of as a, I want to and so that), and the user story description information is shown in Table I.

TABLE I
The user stories set in Book Factory.

US ID	***Dimension***	***User Story***
US1	Role Feature Benefit	As an owner I want my clients to be able to place orders online so that the customer-friendliness of our services increases.
US2	Role Feature Benefit	As a client I want to complete an order so that I can place it online.
US3	Role Feature Benefit	As a client I want to fill my online cart with products.
US4	Role Feature Benefit	As a client I want to pay my invoice so that I can complete an online order.
US5	Role Feature Benefit	As a system component I want to calculate the total amount of the order so that the invoice can be paid .
US6	Role Feature Benefit	As a system component I want to pay my order online so that my invoice is paid .
US7	Role Feature Benefit	As a system component I want to process payments on the Ogone-payment platform so that the payment is secured.

Next, we describe the process of this case according to the method proposed in section III.

A. Node identification from BookFactory

After we input the user stories in Table I and execute the process of node identification, we extract the nodes with *role*, *intention* and *quality* type. Fig. 2 shows the results of this process. Here, nodes with *intention* type are graphically represented as ovals, nodes with *role* type are represented with circle, while qualities are represented as more curved cloud-like shapes.

B. Merging similarity nodes

Second, the BERT model-based approach generates node embedding for each node; then any two node embeddings are calculated using the cosine similarity algorithm to derive a similarity score; the pairs of nodes with our score greater

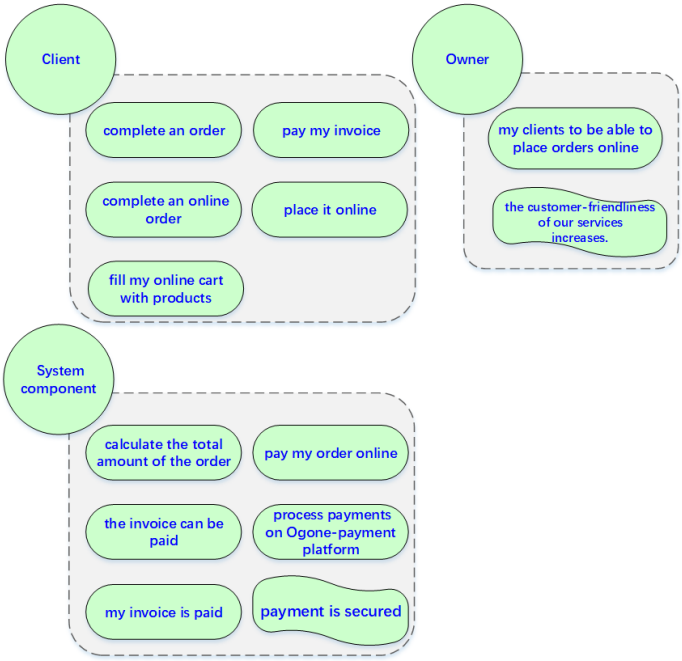


Fig. 2. Identifying iStar nodes of the run example

than a threshold are merged. Here, we list the merged nodes together for viewing ease. The result of this step is shown in Fig. 3.

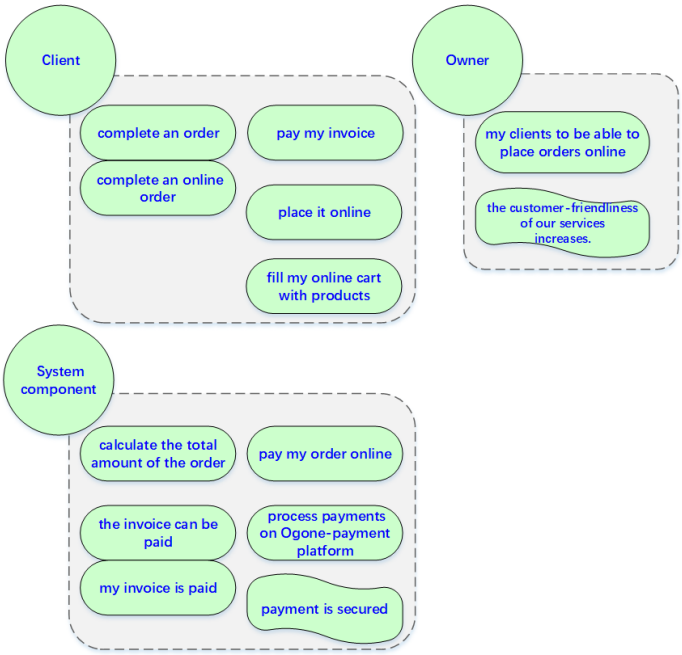


Fig. 3. Merging similarity nodes of the run example

C. Edge identification

On the basis of node identification and merging similarity nodes, we identify the edges between nodes according to rules, as shown in Fig. 4. Here, the *contribution* edges are represented graphically via a dotted line connecting the node type that is qualifies. The *refinement* edges are represented graphically via a solid arrow directed connecting the node type that is merged node.

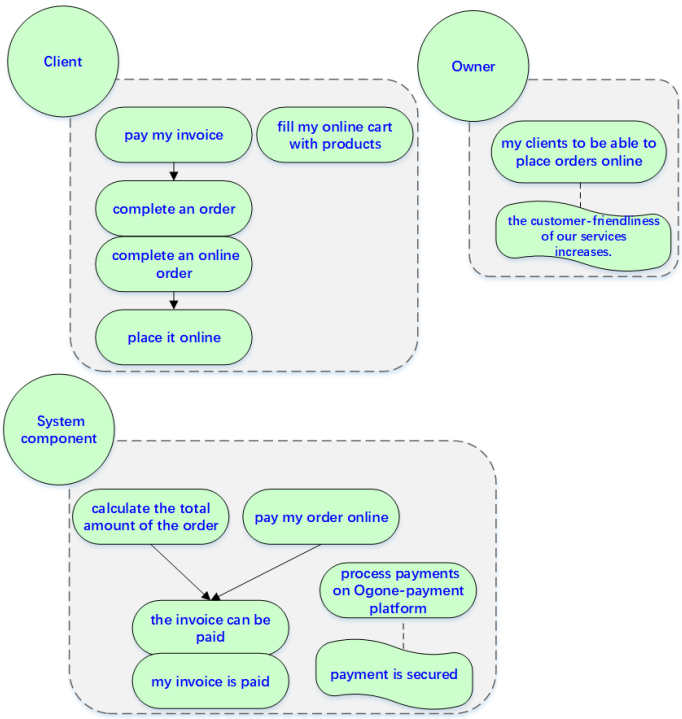


Fig. 4. Identifying iStar edges of the run example

V. Evaluation with Experiment

We evaluate the effectiveness of our proposed approach in both *quality* node extraction and similarity node merging. 3 sets of user stories from different fields (game, business and education) are used to evaluate our approach. These 3 data sets are first manually labeled and then used to evaluate the performance of the proposed approach.

A. Dataset

a) GA: GA is from Ben-Gurion University of the Negev and is used for model building experiment [22]. The data set contains 21 user stories with correct syntax, which mainly describes the needs of setting up games and playing games in the development of game products.

b) WebCompany: webCompany comes from a Dutch company which makes custom web business applications [23]. The data set contains 79 user stories with correct syntax, which mainly covers the development of a web application focused on interactive story telling in 2014.

c) BADcamp: The BADcamp case study is obtained from the public user story requirements data set [24]. It is composed of 70 user stories with correct syntax. It mainly describes the needs of the educational platform in five scenarios: scheduled meeting, training course, sponsorship, blog and session.

B. Manually labeled datasets

For these 3 data sets, we conducted two sets of labeling experiments. One is for quality requirements, and the other is for finding similar pairs of user stories. The participants in these labeling experiments consist of 4 graduate students with software engineering related knowledge. Each student has one semester of software engineering class experience.

In order to ensure the availability of the labeled results, in each group of the labeling experiments, we divided the participants into two groups. The labeling process is as follows:

- Divide the data to be labeled into two parts;
- Divide the 4 students into two groups;
- Each group separately labels a part, and each student of the group labeled all the data in the part assigned to the group;
- Each group separately reviews the inconsistent data labeled by other group, and form a final labeled data set.

When labeling similar pairs, since there are a total of 170 user stories in the 3 data sets, 57630 pairs to be labeled will be generated. After preliminary research, we found that pairs with a similarity lower than 0.5 basically do not have the same semantics. So, in the labeling stage, we only let the students label pairs with a similarity greater than 0.5 and let them determine if the pairs are similar. A similarity greater than 0.5 have 443 pairs, then there were conflicting 71 pairs in the first round, then all conflicting pairs were identified in the second round, we finally obtained 49 similar pairs.

C. Metrics

We use precision, recall, and F1-score to evaluate the effectiveness of *quality* node identification and the merging node identification. The precision is used to evaluate the correctness of the node identification. The recall is used to measure the coverage. The F1-score is used to balance the accuracy and recall of the model. Equation 2, Equation 3, and Equation 4 are used to measure the precision, recall and F1-score, respectively. Where, TP represents true positive which is the nodes identified by automated approach, FP represents false positive which is the nodes identified by automated approach but not by labeled, and FN represents false negative which is the nodes identified by labeled but not by automated approach.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

D. Experiment Result and analysis

Table II shows the precision, recall and F1 score to identify quailty node and similarity pairs for each of the 3 user story sets. For quality nodes identification, the average precision, recall and F-value of our method based on quality words are 76.5%, 73.3%, and 72.1%, respectively. BADcamp data set has a higher precision (100%) than other data sets. The reason is that it only uses the “quickly” and “easily” quality words in non-function description. In other data sets, there are still some quality nodes are not found, the reason is the keyword list extracted by this paper less the related quality works and need to be expanded.

For similar node identification, we summarize the precision, recall and F-1 score of manually labeled semantically identical pairs with different thresholds (0.5-1, step size is 0.1) in the data sets, as shown in Figure 5. The result shows that we can select nodes with a similarity greater than 0.8 to merge.

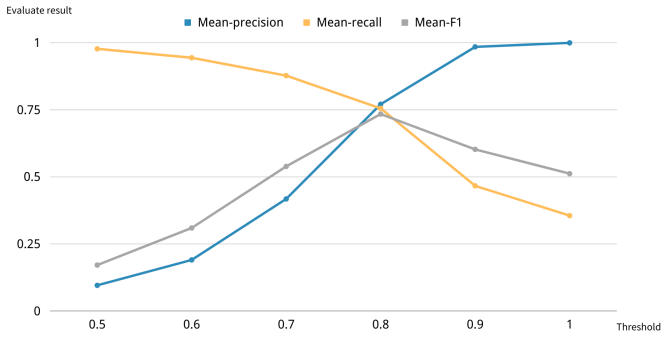


Fig. 5. Similarity result with different thresholds in three datasets

We merge node pairs with a threshold greater than 0.8, and the results of precision, recall and F-1 score are 77.1%, 75.6%, and 73.4%, respectively (the details shown in Table II). GA data set has a higher recall (100%) than other data sets. The reason is that similar sentences of manually label to besides have a sentence to exactly the same, the other sentences in length and the key action and object are the same. Such features can be used to find pairs of similar sentences by our approach. Then webcompany data set has a higher precision (89.7%) than other data sets. The reason is that it has several same nodes in this dataset. Then BADcamp data set has a worst recall(40%) than other data sets. The reason is that the similarity score of sentence pairs are relatively close, and the high threshold value leads to some nodes not being discovered.

VI. Conclusions and Future Works

Software requirements are usually expressed in natural language (such as user stories). Although the text is readable, it is difficult to provide an overall view of the most relevant entities and relationships. Especially in the case of a sharp increase in requirements, it is becoming more and more difficult to get the whole in mind. Model-driven development uses model to create a product and form a whole perspective to observe user needs. The iStar model is a widely used goal-oriented model. Its concepts can be aligned with user stories.

To construct iStar model from user stories, the nodes and edges in iStar model need to be identified from user stories. However, since the user stories provide by different stakeholders are scattered, the relationships between user stories

TABLE II
EFFECT OF USER STORY ANALYSIS OF THREE DATASETS

Dataset	Quality nodes						Similarity of node pairs					
	TP	FP	FN	Precision	Recall	F1-score	TP	FP	FN	Precision	Recall	F1-score
GA	2	1	2	66.6%	50%	57.1%	4	2	0	66.6%	100%	80%
webCompany	7	4	0	63%	100%	77.7%	26	30	4	89.7%	86.7%	88.1%
BADcamp	7	0	3	100%	70%	82.3%	6	2	9	75%	40%	52.1%

are difficult to identify. This paper focuses on the refinement and contribution relationships between user stories. A node-merging based approach is proposed to identify the potential refinement relationships. A quality word list is built to find the non-function user story and to identify the contribution edges. 3 data sets are used to evaluate the effectiveness of the proposed approach.

As for our next step work, we plan to develop a prototype tool that implements our proposed framework. In addition, we will conduct extended experiments on more data sets to verify the effectiveness of the proposed approach.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No.62162051), the Project of Beijing Municipal Education Commission (No.KM202110005025), the Natural Science of Foundation of Inner Mongolia Province (No.2021MS06024) and Inner Mongolia Normal University Graduate Students'research Innovation fund(CXJJS21158).

REFERENCES

[1] B. Ramesh, C. Lan, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," in *Information Systems Journal*, 2010, pp. 60–67.

[2] M. Aoyama, "Agile software process and its experience," in *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, 1998, pp. 3–12.

[3] M. Cohn, "user story applied for agile software development," in *Addison-Wesley Professional*, 2004, p. 304.

[4] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, and S. Poelmans, "Building a rationale diagram for evaluating user story sets," in *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, 2016, pp. 1–12.

[5] J. Lin, H. Yu, Z. Shen, and C. Miao, "Using goal net to model user stories in agile software development," in *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014, Las Vegas, NV, USA, June 30 - July 2, 2014*. IEEE Computer Society, 2014, pp. 1–6.

[6] R. Mesquita, A. Jaqueira, M. Lucena, C. S. Filho, and F. M. R. Alencar, "Us2startool: Generating i* models from user stories," in *Proceedings of the Eighth International i*Workshop, iStar 2015, in conjunction with the 23rd International Requirements Engineering Conference (RE 2015), Ottawa, Canada, August 24-25, 2015*, ser. CEUR Workshop Proceedings, vol. 1402. CEUR-WS.org, 2015, pp. 103–108.

[7] T. Gune and F. B. Aydemir, "Automated goal model extraction from user stories using nlp," in *IEEE Requirements Engineering Conference*, 2020, pp. 382–387.

[8] F. Dalpiaz, X. Franch, and J. Horkoff, "istar 2.0 language guide," in *Software Engineering*, 2016, https://arxiv.org/abs/1605.07767/.

[9] C. Wang, C. Wu, T. Li, and Z. Liu, "A preliminary framework for constructing istar models from user stories," in *iStar Workshop*, 2021, pp. 35–41.

[10] A. Jaqueira, M. Lucena, F. M. R. Alencar, J. B. de Castro, and E. Aranha, "Using i * models to enrich user stories," in *iStar*, 2013, pp. 55–60.

[11] F. A. Mihany, H. Moussa, A. Kamel, E. Ezzat, and M. Ilyas, "An automated system for measuring similarity between software requirements," in *Proceedings of the 2nd Africa and Middle East Conference on software engineering*, 2016, pp. 46–51.

[12] G. Navarro, "A guided tour to approximate string matching," in *ACM Computing Surveys*, vol. 33, no. 1, 2000, p. 31–88.

[13] P. Jaccard, "Etude comparative de la distribution florale dans une portion des alpes et des jura," in *bulletin del la societe vaudoise des sciences naturelles*, vol. 37, no. 142, 1901, pp. 547–579.

[14] A. F. de Araújo and R. M. Marcacini, "Re-bert: automatic extraction of software requirements from app reviews using bert language model," in *Requirements Engineering*, 2021, p. 1321–1327.

[15] T. Li, "Identifying security requirements based on linguistic analysis and machine learning," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2017, pp. 388–397.

[16] A. Ahmad, C. Feng, K. Li, S. M. Asim, and T. Sun, "Toward empirically investigating non-functional requirements of ios developers on stack overflow," in *IEEE Access*, vol. 7, 2019, pp. 61 145–61 169.

[17] M. S. Kumar and A. Harika, "Extraction and classification of non-functional requirements from text files: A supervised learning approach," in *PSYCHOLOGY AND EDUCATION*, vol. 57, 2020, pp. 4120–4123.

[18] A. Mahmoud and G. Williams, "Detecting, classifying, and tracing non-functional software requirements," in *Requirements Engineering*, vol. 21, no. 3, 2016, pp. 357–381.

[19] https://huggingface.co/all-MiniLM-L12-v2.

[20] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *ICLR*, 2017.

[21] Y. Wautelet, M. Velghe, S. Heng, S. Poelmans, and M. Kolp, "On modelers ability to build a visual diagram from a user story set: A goal-oriented approach," in *Requirements Engineering: Foundation for Software Quality*, 2018, pp. 209–226.

[22] F. Dalpiaz, "Form b-eng.pdf.utrecht university. dataset." https://doi.org/10.23644/uu.11605254.v1, 2020.

[23] M. Robeer, G. Lucassen, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via nlp," in *Requirements Engineering Conference*, 2016, pp. 196–205.

[24] F. Dalpiaz, "Requirements data sets (user stories)," https://data.mendeley.com/datasets/7zbk8zsd8y/1, 2018.

KEMA: Knowledge-Graph Embedding Using Modular Arithmetic

Hussein Baalbaki
ISEP
Sorbonne University
Paris, France
hussein.baalbaki@ext.isep.fr

Hussein Hazimeh and Hassan Harb
Faculty of Sciences
Lebanese University
Beirut, Lebanon
hussein.hazimeh@ul.edu.lb, hassan.harb.1@ul.edu.lb

Rafael Angarita
ISEP
Sorbonne University
Paris, France
rafael.angarita@isep.fr

***Abstract*—Knowledge graph is a knowledge representation technique that helps representing entities and relations in a machine understandable way. This promising trend suffers from the problem of incompleteness that was best solved by link prediction. Indeed, link prediction is the most successful method for understanding the structure of the large knowledge graphs. Knowledge graph embedding KGE is one of the best link prediction methods. Its effectiveness is mainly affected by the accuracy of learning representations of entities and relations. In this paper, we propose a new knowledge graph embedding model called KEMA (Knowledge-graph Embedding using Modular Arithmetic). KEMA has the ability to represent simple and complex relations in an efficient way. Consequently, this allows our model to outperform the majority of the existing models. Mainly, KEMA depends on representing the relations in a knowledge graph by modular arithmetic operations applied between entities. Experimental results on multiple benchmark knowledge graphs verify the accurate representation, low complexity and scalability of KEMA.**

Keywords; Knowledge Graph Embedding, Knowledge Graphs, Link Prediction, Reasoning, Modular Arithmetic.

I. Introduction

Knowledge graph (KG) rises recently as one of the best ways for knowledge representation. We have seen the construction of many KGs of different sizes, domains, and coverage, like Freebase [1], Yago [2], and WordNet [3]. KG is a multi-relational graph built of nodes representing real world entities such as objects and events. These entities are connected by edges representing the relations and interactions between them. KG is represented by a set of triplets that shows the relations linking the entities. KG has proven to be effective in many real-world applications like recommender systems [4], natural language processing [5], and question-answering [6].

Although a KG may consist of a huge number of entities and relations, it is usually incomplete. It is impossible for a KG to cover every single entity or relation in whole world, no matter how huge this KG is. This is called *the completeness problem* of KG. This challenge represents one of the main issues facing KGs that researchers are working to solve. Link prediction emerges as efficient way to overcome the KG completeness problem. Subsequently, link prediction is used to predict not only the existence of a relation between two entities, but also the specific type of this relation. However, these predictions are infeasible using traditional methods. So the need for novel link prediction approaches like Knowledge graph embedding arises. Knowledge graph embedding (KGE) methods have proven to be very effective applied in link prediction. KGE embeds a KG into a continuous vector space while preserving certain information of the graph. Generally, KGE replaces any object (entity, relation, ..) with a vector of continuous numbers holding this object semantics. Mainly, KGE models differ in how these numeric vectors are used. and divided into three categories accordingly. The first one is the Translational that consider relations as a motion between entities. The second category is Tensor factorization that uses tensors for embedding vectors processing. The third category is the Neural network, in which the embedding vectors are fed to neural networks for training.

In this paper, we introduce a novel knowledge graph embedding model based on the modulus operation called as KEMA. Indeed, KEMA adopts a new way for dealing with embedding vectors away from translational, tensor factorization, and neural networks. Our model relies on the modular arithmetic mathematical operation. Since modular arithmetic is an equivalence relation, it helps handling different types of knowledge graph relations such as symmetry, inversion, and composition. Moreover, KEMA can deal with relations of complex mapping patterns like one-to-many, many-to-one, and many-to-many. To prove the effectiveness of our proposed model, we evaluate KEMA on a set of knowledge graph benchmark datasets including FB15k-237 [7] and WN18RR [8], and we compared the results to state-of-the-art approaches.

The rest of the paper is organized as follows. Section II gives an overview about different knowledge graph embedding models existing in the literature. Section III explains the modular arithmetic mathematical operation and its working way. Section IV presents KEMA, our proposed model. The obtained simulation results are exposed in Section VI. Finally, we conclude the paper and provide directions for future work in section VII.

II. Related Work

In the literature, we can distinguish between three types of knowledge graph embedding categories [9]: (1) Translation-

DOI reference number: 10.18293/SEKE2022-036

based models; (2) Tensor factorization-based models; (3) Neural network-based models.

A. Translation-based models

The first translational-based model is called TransE [10]. Typically, TransE considers the relation between two entities as a translation operation in embedding space that starts by the head entity and leads to the tail. A score function f is used to compute the authenticity of the given triplet (h, r, t): $f_r(h, t) = |h+r-t|$. Although TransE shows a high efficiency when applied in large-scale knowledge graph embedding, it still struggle when dealing with complex relations such as $1-N, N-1$, and $N-N$. In order to overcome such challenge, the authors of [11] proposed an extension of TransE called TransH. Indeed, TransH assigns a hyper-plane to each relation, so that the heads and the tails of this relation are projected to. This model enables each entity to have different embedding representations depending on the relation involved in. In [12], TransR focuses on the entities carrying various semantic meanings. As a result, TransR expands the relation-specific hyper-planes concept of TransH to relation-specific spaces. Compared to transE and transH, transR makes a significant performance improvement but still has some gaps. One TransR weakness is sharing one projection matrix by both head and tail. The attributes of the same entity may differ according to its position in a relation (head / tail). To solve this problem, TransD [12] solves this problem by creates two different projection matrices for each single relation. The first matrix is used for head projection and the other is for the tail projection. Lastly, the authors of [13] proposed RotatE that represents the relation between two entities as rotation motion that starts from the head entity and ends by the tail.

B. Tensor factorization-based models

The idea behind tensor factorization-based models is based on representing all the triplets of a knowledge graph in a 3D binary tensor X. Two dimensions of X are of size n, the number of the entities in KG, while the third dimension is of size m, the number of relations in KG. S_r is $(n*n)$ slice of X that represents the relation r. The index $S_r[head][tail]$ of the slice S_r is filled with 1 if the relation between the head and the tail entities holds, otherwise it is filled by 0. The obtained tensor X is then broken down using factorization into a set of embedding matrices that are assigned to entities and relations. RESCAL [14] uses $Rank-d$ factorization to obtain three matrices:

$$X = ARA^T, \tag{1}$$

where A is a 2D matrix carrying the semantics of the KG entities, A^T is its transpose, and R is the 3D embedding matrix of the relations in KG. Accordingly, the scoring function of a given triplet (h, r, t) is calculated as follow:

$$f_r(h, r) = h^T m_r t, \tag{2}$$

where h^T is the transpose of h embedding vector, m_r is a 2D slice of R holding the relation r embedding matrix, and t is the tail embedding vector. To reduce the calculation complexity, a simplified version of RESCAL called DistMult is proposed in [15].In DistMult, any relation matrix m_r, obtained after factorization is considered to be diagonal with a score function:

$$F = h^T diag(m_r) t. \tag{3}$$

This step not only facilitates calculations, but it also shows improvement in terms of performance. Moreover, in order to capture the pairwise compositional representations of entities, HolE is proposed in [16]. Basically, HolE relies on a circular correlation with the following score function

$$f_r(h, t) = r^T (h \star t) \tag{4}$$

HolE reduces the composite representation complexity compared to tensor product. Lastly, ComplEx [17] was proposed as extension of DistMult, where it closes the gap resulting from the inability of DistMult to deal with asymmetric relations by integrating a complex space as follows:

$$fr(h, t) = Re(h_t * diag(r) - t) \tag{5}$$

where Re(·) denotes the real part of a complex value, and t represents the complex conjugate of t.

C. Neural network-based models

Neural networks are used for expressing complex nonlinear projections. They become recently a hot topic used to embed a knowledge graph into a continuous feature space. Particularly, Semantic Matching Energy SME [18] is a neural network-based model that calculates the energy of a given triplet (h, r, t) by applying two projection matrices, M_{left} is applied for the head h and the relation r embedding vectors, while M_{right} is applied for relation r and the tail t embedding vectors. Then, the results are given to a fully connected layer that returns the score of the semantic matching energy. ConvKB [19] is another model that uses convolutional neural network (CNN) to capture the latent semantic information in the triplets. In such model, the embedding vectors of the elements of a triplet are combined to form a matrix. Then, the matrix is fed to a convolution layer to produce multiple feature maps. Finally, these feature maps are concatenated and projected to a score that is used to estimate the authenticity of the triplet. Neural Association Model NAM [20] utilizes a deep neural network structure to represent a KG. After representing each element of a triplet by embedding vector, it concatenates the head entity vector and the relation vector to a single vector. The single vector is then fed to the next layer. Finally, NAM calculates the score by applying the output of the last hidden layer z^L with the tail embedding vector t:

$$f_r(h, t) = \sigma(z^L t) \tag{6}$$

where $\sigma(\cdot)$ is a sigmoid activation function.

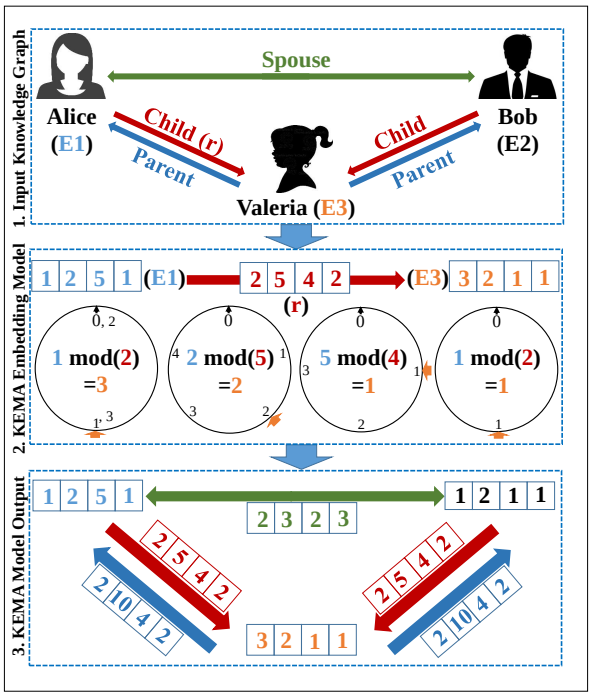


Fig. 1. KEMA architecture.

III. Our Embedding Model: KEMA

Knowledge graph embedding is the process of representing the entities and relations of a given knowledge graph using numerical vectors. In this way, mathematical operations can be applied to these vectors in order to help studying and predicting the links connecting the entities. In this paper, we introduce a new embedding KG approach called KEMA. Indeed, KEMA does not follow any of the classifications detailed in the related work section. It relies on one simple mathematical operation called modular arithmetic. The objective of KEMA is to predict the missing links connecting the entities of a given knowledge graph. As shown in Fig. 1, first, the input knowledge graph layer receives a knowledge graph as input. Then, KEMA embedding layer processes the knowledge graph components and embeds it to a low dimension continuous space. Finally, KEMA output layer returns a representative numerical vector for every entity and relation in the KG. These representative numerical vectors are then used for predicting links between KG entities.

A. Modular arithmetic

Modular arithmetic is a system of arithmetic that replaces all the numbers by their remainders of its division to a fixed number. Subsequently, the fixed number is an integer called "modulus". In modular arithmetic, every value of modulus m can be considered as a representative space $mod(m)$. In this space, we can represent every integer i by its remainder r resulting from its division by m, as shown in Equation (7). Since all the remainders of the division by m are smaller than m itself, then all the representative integers in $mod(m)$ space are smaller than m (Equation (8)).

$$i = m * x + r \iff r = i \ mod(m), \quad \forall \ i \in \mathbb{Z}, \tag{7}$$

$$r = i \ mod(m) \iff r \in]-m, m[\tag{8}$$

where $m, r, x \in \mathbb{Z}$

Indeed, an important example to illustrate the process of the modular arithmetic in this paper is 12-hour clock. In such example, the modulus value is 12 and the day is divided into two 12-hour periods. Whenever the hours count exceeds 12, it wraps around, and returns the remainder value.

Usually, modular arithmetic produces a set of integers having the same remainder when dividing by m. These integers are considered equal in the $mod(m)$ space, and are said to be congruent ($\equiv$), as shown in Equation (9).

$$\begin{aligned} x \ mod(m) &= y \\ z \ mod(m) &= y \\ x \ &\equiv \ z \end{aligned} \tag{9}$$

B. KEMA Embedding Model

The novel idea behind KEMA is to represent the relation between two entities through modular arithmetic operation. In other words, the tail embedding vector is considered to be the projection of the head embedding vector in the modular arithmetic space of the relation embedding vector, as shown in Equation (10).

$$t \ = \ h \ mod \ (r) \tag{10}$$

where h, t, and r represent the embedding vectors of the head, the tail and the relation respectively.

The second layer of our model is called KEMA embedding model (see Fig. 1), and it shows the way the embedding vectors are assigned for entities and relations of a given KG. KEMA starts by assigning random vectors for entities and relations. Then, it modifies these vectors in a way it satisfies the score function shown in Equation (10). First, every index $E_h[j]$ in the vector of the head entity E_h is subjected to modular arithmetic operation of modulus $r[j]$, the j-th index of relation r. Then, the vector of numbers obtained from this operation is assigned to the tail entity E_3 of the relation r.

C. Types of Relations

Despite the simplicity of the calculation process used in KEMA, it has proved to be highly effective and accurate compared to other models. This simplicity can also be seen in the low complexity of both training and prediction processes. As well as simple relations, KEMA can effectively handle complex relations of KG such as 1-N and N-N.

1) Simple Relations: Simple relation is that connecting no more than two entities, the head and the tail. According to the existing literature, three types of simple relation patterns are very important: symmetric, inverse, and composed patterns. All these patterns are covered by KEMA embedding model as follows:

- *Symmetric Relation*: This relation, switching between the head and the tail entities of a relation is possible. A relation r is said to be symmetric, if $\forall x, y \in E$, the set of entities

$$(x, r, y) \implies (y, r, x) \tag{11}$$

- *Inverse Relation*: A relation r_2 is said to be inverse of relation r_1 whenever r_1 and r_2 have opposite directions connecting the same entities. A relation r_2 is the inverse of r_1, if $\forall x, y \in E$, the set of entities in KG

$$(x, r_1, y) \implies (y, r_2, x) \tag{12}$$

- *Composed Relation*: A relation r is said to be composed, if it can be broken down into two relations or more. A relation r is a composed relation, if $\exists\ r_1, r_2 \in R$, the set of relations in KG

$$(x, r, z) + (z, r, y) \implies (x, r, y) \tag{13}$$

Where $x, y, z \in E$, the set of entities.

For Modular Arithmetic

Let $a, b, n \in \mathbb{Z}$ such that:
$a \equiv b(mod\ n)$
$\implies a - b = kn$, for some $k \in \mathbb{Z}$
$\implies b - a = (-k)n$ and $-k \in \mathbb{Z}$
$\implies b \equiv a(mod\ n)$
Thus modular arithmetic is a symmetric relation.

Let $a, b, n \in \mathbb{Z}$ such that:
$a \equiv b(mod\ n)$
$\implies a - b = kn$, for some $k \in \mathbb{Z}$
$\implies b - a = (-n)k$ and $-n \in \mathbb{Z}$
$\implies b \equiv a(mod\ k)$
mod(n) is inverse to mod(k).
Thus modular arithmetic is an inverse relation.

Let $a, b, n, c \in \mathbb{Z}$, such that:
$a \equiv b(mod\ n)$ and $b \equiv c(mod\ n)$.
then $a = b + kn, k \in \mathbb{Z}$ and $b = c + hn, h \in \mathbb{Z}$.
$a = b + kn$
$\implies a = (c + hn) + kn$
$\implies a = c + (hn + kn)$
$\implies a = c + (h + k)n,\ h + k \in Z$.
Hence $a \equiv c(mod\ n)$.
Thus modular arithmetic is a composed relation.

2) KEMA Complex Relations: The majority of the models proposed in the literature can deal with simple relations between KG entities, i.e 1-to-1 relations. However, relying on simple relationships to build knowledge is impractical. Contrarily, KEMA has the ability to handle both simple and complex relationships.

Giving a 1-to-N relationship r, the set of tails $T = \{t_1, t_2, ..t_N\}$ can share a unique head whenever all these tails are congruent in $mod(r)$ space, Equation (14). Similarly, the set of heads $H = \{h_1, h_2, ..h_N\}$ can share the same tail whenever all these heads are congruent in $mod(r)$ space, Equation (15). Moreover, KEMA allows the representation of N-to-N complex relationships, in which one relation can have several heads and tails at once, by combining both equations (14) and (15)

$$t_1 \equiv t_2\ ..\ \equiv t_N \iff h = t\ mod\ (r),\ \forall\ t \in T, \tag{14}$$

$$h_1 \equiv h_2\ ..\ \equiv h_N \iff t = h\ mod\ (r),\ \forall\ h \in H, \tag{15}$$

For Modular Arithmetic

Let $a, b, n, c \in \mathbb{Z}$ such that:
$a \equiv b(mod\ n)$

then $a = b + kn, \quad k \in \mathbb{Z}$
$\implies a = b + (k - c + c)n$
$\implies a = b + cn + (k - c)n, \quad k - c \in Z$.
Hence $a \equiv b + cn(mod\ n)$ and $a \equiv b(mod\ n)$

Thus modular arithmetic holds for 1-N relations.

Given the 1-N relation:

$$a \equiv b + cn(mod\ n) \text{ and } a \equiv b(mod\ n)$$

Since modular arithmetic is symmetric relation, then:

$$b + cn \equiv a(mod\ n) \text{ and } b \equiv a(mod\ n)$$

Thus modular arithmetic can represent N-1 relations.

By combining the 1-N and N-1 modular arithmetic relations, we conclude its ability to represent N-N relation, and thus modulus can represent all the complex relation patterns.

TABLE I
EMBEDDING VECTORS OF ENTITIES AND RELATIONS

Entity	Embedding vector	Relation	Embedding vector
Bob	[1,2,1,1]	$Spouse$	[2,3,2,3]
$Alice$	[1,2,5,1]	$Child$	[2,5,4,2]
$Valeria$	[3,2,1,1]	$Parent$	[2,4,10,2]

D. Analytical example

In this section, we illustrate an example to show the effectiveness of KEMA in terms of representing simple and complex relations. According to the input knowledge graph layer shown in figure 1, the sub graph shows the relations between three entities. Table I shows the embedding vectors that KEMA assigned to every entity and relation.

The relation "$Spouse$" is an example of the symmetric relation. In Fig. 1, the output layer of KEMA shows that this relation holds in both directions. Moreover, in Table II, the first row shows that the tail of the relation "Spouse" with head entity "Bob" is "Alice". On the other hand, the second row represents the opposite direction, where the tail of the relation "Spouse" with head entity "Alice" is "Bob".

TABLE II
SYMMETRIC RELATION EXAMPLE

Head	Relation	Tail (result)
[1,2,1,1]	[2,3,2,3]	[1,2,5,1]
[1,2,5,1]	[2,3,2,3]	[1,2,1,1]

TABLE III
INVERSE RELATIONS EXAMPLE

Head	Relation	Tail (result)
[1,2,5,1]	[2,4,10,2]	[3,2,1,1]
[3,2,1,1]	[2,4,5,2]	[1,2,5,1]

In Fig. 1, the relations "Child" and "Parent" show the inversion pattern of simple relations. In the first row of the Table III, "Alice" is the head of the relation "Child" while "Valeria" is its tail. In the opposite direction, the second row shows the relation "Parent", where "Valeria" is the head and "Alice" is the tail. Then the relations "Parent" and "Child" are said to be inverse.

Furthermore, Fig. 1 shows that the relation "Spouse" is a composed relation. Fig. 1 shows that the relation "Child" of head "Alice" and tail "Valeria", followed by the relation "Parent" of head "Valeria" and tail "Bob", can be replaced by the relation "Spouse" having the same head as "Child", and the same tail as "Parent".

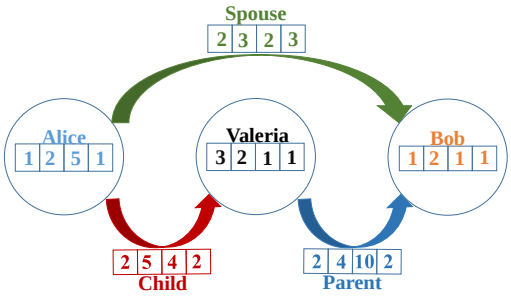


Fig. 2. Composed relation example.

Indeed, the strength of KEMA in terms of representing complex relations is shown through the relations "Parent" and "Child" in Fig. 1. Since "Parent" has one head which is "Valeria", and two tails which are "Bob" and "Alice", then it is a complex relation of $1-N$ mapping pattern. Whereas "Child" relation has two heads "Bob" and "Alice" connected to one tail "Valeria", representing N-1 pattern.

IV. SIMULATION RESULTS

In this section, we will show the experiment setting to implement our model followed by the discussion of the obtained results.

A. Experimental setting

To evaluate our model, we implemented KEMA using Ampligraph python library. Then, we compare it to the state-of-the-art models on two commonly used benchmark datasets: WN18RR, and FB15K-237.

- WN18RR is a subset of WordNet, a KG that clusters words into synonym groups and features lexical relationships between words. It consists of 40,943 entities, 11 relation types, and 93003 triples. WN18RR contains symmetry, antisymmetry and composition relation patterns. The main pattern is the symmetry since almost each word has a symmetric relation in WN18RR, e.g., $also-see$ and $similar-to$ [13].
- FB15k-237 is a subset of Freebase, a large knowledge graph that stores general knowledge facts. It consists of 14951 entities, 237 relation types, and 310116 triples. The main patterns of the relation in FB15k are symmetry, antisymmetry and composition [13].

To train and evaluate our model, we need to use negative triples, which are not available in both WN18RR and FB15K-237. As a result, we used to corrupt the positive triples of the datasets to generate negative samples for each positive. This is what is called the local closed world assumption. That is, for a triple, we randomly replace the entity in the subject or the object position by another, but not both at once.

We applied three tests to evaluate the performance of link prediction of KEMA. These tests rely on ranking each positive test triple against all its generated negatives according to its score. The first test is Mean Rank (MR) which is calculated as follow:

$$mean(rank_t)\ \forall t \in T \tag{16}$$

with T is the set of positive test triples, and $rank_t$ is the rank of triple t against its negatives.

The second evaluation test is Mean Reciprocal Rank (MRR). It is similar to MR, but it uses the reciprocal rank of a triple instead of its rank, what make it less sensitive to outliers [16]:

$$mean(1/rank_t)\ \forall t \in T \tag{17}$$

The last evaluation test is Hits@N, which counts the test triples having a rank less than or equal to N.

$$\sum t_N,\ where\ t_N \in T, rank_{t_N} >= N \tag{18}$$

B. Main Results

In our simulation, we compared KEMA to several state-of-the-art models including TransE [10], DistMult [15], ComplEx [17], ConvE [8], and RotatE [13]. We show the efficiency of our proposed model inferring the relation patterns for the task of predicting missing links. Tables IV shows the results of the evaluation tests of our model and the state-of-the-art models based on WN18RR and FB15K-237 datasets respectively.

FB15K-237 dataset contains symmetry, anti-symmetry and composition relation patterns. The main pattern in this dataset is the composition [13]. The domination of the composition pattern can be inferred from the results shown in Table IV. So that Table IV shows that the model TransE, representing composition and anti-symmetry patterns, outperforms ComplEx model representing symmetry and anti-symmetry patterns.

TABLE IV
RESULTS OF MODELS EVALUATION ON WN18RR AND FB15K-237 DATASETS

	WN18RR					FB15K-237				
Model	MR	MRR	Hits@1	Hits@3	Hits@10	MR	MRR	Hits@1	Hits@3	Hits@10
TransE	3384	0.226	-	-	0.501	357	0.294	-	-	0.465
DisMult	5110	0.43	0.39	0.44	0.49	254	0.241	0.155	0.263	0.419
ComplEx	5261	0.44	0.41	0.46	0.51	339	0.247	0.158	0.275	0.428
ConvE	4187	0.43	0.40	0.44	0.52	244	0.325	0.237	0.356	0.501
RotatE	3340	0.476	0.428	0.492	0.571	177	0.338	0.241	0.375	0.533
KEMA	**1898**	**0.477**	**0.442**	**0.486**	**0.543**	**244**	**0.311**	**0.223**	**0.342**	**0.486**

Furthermore, Table IV shows the superiority of KEMA over TransE, DistMult, and ComplEx in all the tests, due to its ability to perfectly represent composition and symmetry patterns. On the other hand, ConvE and RotatE surpass KEMA due to its ability to represent all the relation patterns of FB15K-237, which is composition, symmetry, and anti-symmetry.

Table IV shows the results of the evaluation tests of KEMA and the state-of-the-art models on WN18RR dataset. Similar to FB15K-237 dataset, WN18RR contains composition, symmetry and anti-symmetry relation patterns. Symmetry is the dominating pattern in this dataset. This conclusion can be easily inferred from the Table IV. So that the model DistMult, representing only symmetric relations, performs better than TransE representing both anti-symmetry and composition patterns. On WN18RR dataset, our model outperforms all the state-of-the-art models. Although RotatE and ConvE both represent all the relation patterns contained in WN18RR, KEMA surpasses both models that confirms its high performance.

V. CONCLUSION AND FUTURE WORK

Link prediction is among the most prominent methods that solve the problem of incompleteness of Knowledge graph. It is used to predict the existence and the type of a relation connecting two entities. The more the knowledge graph is well represented, the more the predictions are accurate. In this paper, We proposed a novel knowledge graph embedding model (KEMA) that relies on modular arithmetic operation in representing relations between entities. KEMA applies modular arithmetic to the head entity with modulus equal to the relation vector. The main strength of our model lies in its ability to represent complex relations like one-to-many, in addition to representing symmetry, inverse, and composed simple relations. The results of our experiments show that KEMA outperforms the majority of the existing models in representation accuracy while preserving low level of complexity.

In the future work, we plan to build a complete KEMA framework that contains beside the proposed model a loss function and a suitable negative sampling method.

REFERENCES

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[3] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[4] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[5] B. Yang and T. Mitchell, "Leveraging knowledge bases in lstms for improving machine reading," *arXiv preprint arXiv:1902.09091*, 2019.

[6] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 221–231.

[7] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 2015, pp. 57–66.

[8] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[9] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, p. 750, 2020.

[10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, pp. 1–9, 2013.

[11] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[12] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 687–696.

[13] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," *arXiv preprint arXiv:1902.10197*, 2019.

[14] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.

[15] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[16] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[17] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2071–2080.

[18] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[19] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," *arXiv preprint arXiv:1712.02121*, 2017.

[20] Q. Liu, H. Jiang, A. Evdokimov, Z.-H. Ling, X. Zhu, S. Wei, and Y. Hu, "Probabilistic reasoning via deep learning: Neural association models," *arXiv preprint arXiv:1603.07704*, 2016.

Embedding Knowledge Graphs with Semantic-Guided Walk

Hao Tang, Donghong Liu, Xinhai Xu*, and Feng Zhang*
Academy of Military Science, Beijing, China

***Abstract*—Knowledge graph completion can complete knowledge by predicting missing facts, which is a increasingly hot research topic in knowledge graph construction. Prevalent approaches propose to embed knowledge graphs in a low-dimensional vector space and use these embedding to predict, but they neglect either semantic information or graph structures. We propose a new approach to knowledge graph completion named as ATTWALK, which learns embedding by exploiting both structural and semantic features of a knowledge graph. This is achieved by leveraging a key insight that an entities' embedding is influenced by its multi-hop neighbors', which can be further distinguished by their semantic importance to the entity. ATTWALK orchestrates a two-step workflow by first evaluating neighbors' semantic weights using graph attention networks for each entity, then exploring the entities' local structural features by performing a semantic weight guided walk. We evaluate ATTWALK by conducting extensive experiments, which show that ATTWALK outperforms 12 representative approaches on average across 3 publicly available datasets.**

***Index Terms*—knowledge graph embedding, graph attention networks, random walk**

I. INTRODUCTION

A knowledge graph (KG) is a directed graph which excels at organizing relational facts. It represents factual entities as nodes and semantic relations as edges. For a fact that entity `h` has a relationship `r` with entity `t`, KGs model it as an edge *r* pointing from node *h* to node *t*, which is denoted as a triple (*h*, *r*, *t*). As a structured form to model human knowledge, many large-scaled KGs have emerged as the backbone of many AI related applications such as question answering [1], recommendation systems [2] and intelligent services [3], and become increasingly important nowadays.

Although KGs can be large in size, they are far from complete. This gives rise to the task of automatic KG completion, which aims at predicting missing facts based on existing triples in a KG. A prevalent research direction proposes to map nodes and edges of a KG into distributed representations in a low-dimensional vector space, so as to simplify the prediction while preserving their relations. This is also known as knowledge graph embedding (KGE) and is gaining massive attention recently.

Among all the previous KGE works, facts-based and relation path-based approaches are two main representatives, but they either overlook rich structural features or semantic information of a KG. Facts-based approaches take a KG as a set of triples (i.e., facts). They propose different scoring functions on the embedding of each triple to measure its factual plausibility, and obtain the embeddings by maximizing the function values. Instead, relation path-based approaches compute embeddings considering multi-hop relationships and handle relation paths using composition strategies [4], but the huge number of composed paths brings critical complexity challenges [5]. This gives rise to more feasible approaches of path sampling, which often relies on heuristics or prior knowledge but under-utilizes semantic information in a KG. Such approaches include Node2Vec [6], RelWalk [7] and so on.

Motivated by the above challenges, we propose a new approach named ATTWALK to knowledge graph completion, which learns embedding by exploiting both structural and semantic features of a KG. Unlike using semantic information from textual material [8], we consider using semantic information from the triple itself. ATTWALK is inspired by a point of view in social reference theory [9] that an individual's social role is influenced by his or her personal relational network instead of only direct links, moreover, those entities in this network contribute differently to his or her social role. As an analogy, for a node n in a KG, n's embedding essentially models its role or feature, which is influenced by the *community* (i.e., a sub-graph) c that n is located in, instead of only n's one-hop neighbors, and the influences posed by nodes in c may vary according to their semantic importance to node n. Following this idea, ATTWALK orchestrates a simple two-step workflow: for each node n, its multi-hop neighbors' semantic weights are first evaluated using graph attention network, then the weights are used to guide a random walk starting from n in order to aggregate its multi-hop neighbors' influences and obtain n's embedding.

In summary, the contributions of this paper are threefold: (1) We borrow an idea from sociology and propose a simple but effective approach ATTWALK to KGE. ATTWALK provides a new angle that can exploit both graph structures and semantic relations to embedding KGs; (2) We design and implement a workflow to put the idea of ATTWALK into effects; and (3) We evaluate ATTWALK by conducting extensive experiments. ATTWALK outperforms 12 related approaches on average across three publicly available datasets, which demonstrates the effectiveness of ATTWALK.

II. MOTIVATION

Figure 1 serves as a motivating example throughout this section, which is simplified from a large-scaled KG due to space limit.

DOI reference number: 10.18293/SEKE2022-048
* corresponding author

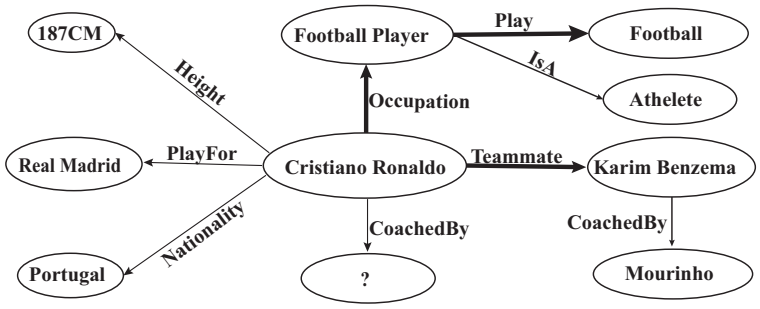


Fig. 1. A KG example describing some real-life relations connected with Cristiano Ronaldo.

We first describe related preliminaries to KGE (Section II-A), then introduce the general ideas of existing works (Section II-B) and our approach (Section II-C) in solving the KGE task. Without loss of generality, let us focus on how these approaches learn the embedding of the specific node *Cristiano Ronaldo* in Figure 1

A. Preliminaries

Knowledge Graph. A knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations respectively. Each triple (edge) (*h*, *r*, *t*) contains a subject entity $h \in \mathcal{E}$, a predicate $r \in \mathcal{R}$, and a tail entity $t \in \mathcal{E}$, denoting head entity *h* has a relation *r* to tail entity *t*. For example, $(\textit{Cristiano Ronaldo}, \textit{Nationality}, \textit{Portugal})$ implies the nationality of *Cristiano Ronaldo*, as is shown in Figure 1.

B. Related Work

Facts based approaches generally take the KG in Figure 1 as a set of 8 triples, one for each edge connecting two nodes. For each triple, they propose a model to describe how the triple holds and a scoring function to measure its plausibility. Embeddings are obtained by maximizing the value of the scoring function. In this way, the semantic of each triple is captured, but the multi-hop neighbors' influence and graph structures are not fully utilized. For example, although *Football* is a 2-hop neighbor of *Cristiano Ronaldo*, it is more semantically relevant than *Cristiano Ronaldo*'s 1-hop neighbor *187cm*. These approaches can be divided into four categories: (i) translation-based models, which consider the translation operation between entity and relation embedding, such as TransE [10] and TransH [10]; (ii) factorization-bsaed models, which assume KG as a third-order tensor matrix, and the triple score can be carried out through matrix decomposition. Such as RESCAL [11], HOLE [12]; (iii) CNN-based models, which employ convolutional neural networks to determine the scores of triples, such as ConvE [13] and ConvKB [14]; and (iv) Graph neural network-based models, which extend convolution operations onto non-Euclidean graph structures, such as RGCN [15], KBGAT [16], EIGAT [17] and CompGCN [18].

By contrast, the path sampling based approaches lay more emphasis on the graph structures around the target node. They perform a truncated random walk starting from node *Cristiano Ronaldo* along the outgoing edges, which results in a path p, then propagate embeddings of nodes in p to the target node's, such as DeepWalk [19], node2vec [6]. However, these approaches learn the embedding of *Cristiano Ronaldo* based on its co-occurrence with nodes in the path p, without utilizing the semantics of edges shown in Figure 1. For example, node *187cm* may pose more influence than node *Football Player* on the embedding of node *Cristiano Ronaldo*, though the former is less semantically relevant than the latter.

C. Our Solution

ATTWALK learns the embeddings considering both semantic information and graph structures of the KG, and conducts a two-step procedure. Step-1: For each node n, ATTWALK first evaluates the importance of n's neighbors, based on the intuition that different neighbors pose different influences to n. For example, to node *Cristiano Ronaldo*, the path *Occupation* $\rightarrow$ *Football Player* should contribute more to its embedding than path *Height* $\rightarrow$ *187cm*, therefore, the former is assigned more weights than the latter. This is achieved using a graph attention network. Step-2: Starting from node *Cristiano Ronaldo*, a truncated walk is performed, but which path to take is guided by the weights of the paths. The sampled paths are then aggregated to capture multi-hop neighbors' influences on node *Cristiano Ronaldo*. In this way, both structural and semantic features are exploited in the embedding. Moreover, the relational importance is automatically learned without prior knowledge.

III. ATTWALK

This section introduces our approach ATTWALK. The general architecture is shown as Figure 2. We detail its critical technical components, the graph attention module and the weighted walk module, in Section III-A and Section III-B, respectively.

A. Graph Attention Network

ATTWALK leverages both entity and relation features in the multi-relation knowledge graph. Because in the knowledge graph, we think relations are as crucial as entities. To better manipulate entity and relation embedding, such as inner product, cross product, and subtraction, we map both of them into the same dimension. Each layer of GAT takes a set of entity features

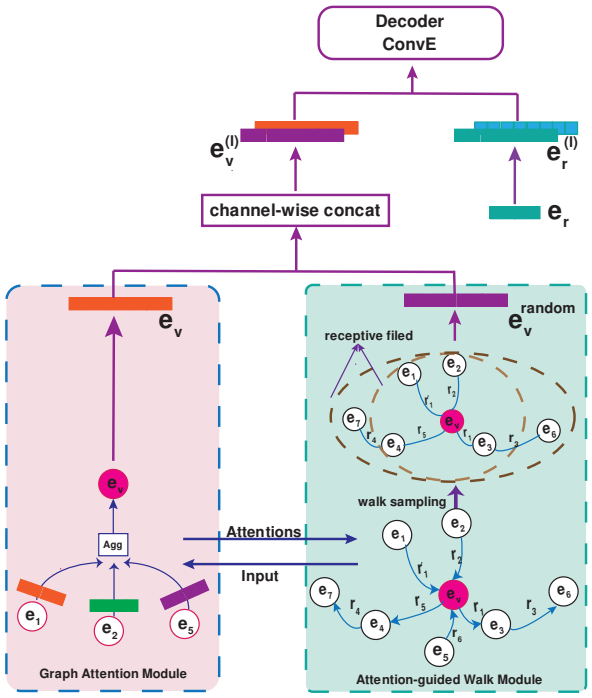


Fig. 2. Overview of ATTWALK architecture.

$e \in \mathbb{R}^{N_e \times P}$ and relation features $r \in \mathbb{R}^{N_r \times P}$ as input, and outputs a new set of entity and relation features: $e \in \mathbb{R}^{N_e \times V}$, $r \in \mathbb{R}^{N_r \times V}$, where the i-th row of e is the embedding of entity e_i and the j-th row of r is the embedding of relation r_j. N_e and N_r is the number of entities and relations, respectively. P and V are the input and output dimension of entity and relation embedding, respectively.

Considering different neighbors may have different importance related to one entity, we perform a shared attention mechanism. We first learn a representation of each triple, for example, *(h_i, r_j, t_k)*, by performing a linear transformation over entity and relation embedding, as shown in Equation 1.

$$c_{ijk} = \mathbf{W}\Phi(\vec{h}_i, \vec{r}_j, \vec{t}_k) \tag{1}$$

where Φ represents operations over entity and relation embedding. Inspired from [10], [20] and [21], we define three kinds of operators, subtracting, multiplying and cross-product. $\mathbf{W}$ is the linear transformation matrix. We also notice that knowledge graphs are directed relational graphs. As shown in Figure 1, *Football Player* is embraced by neighborhood entities *Football* and *Cristiano Ronaldo*, and linked by out-relation *Play* and in-relation *Occupation*. *Football Player* can be either a head entity or a tail entity, so we distinguish the direction of the relations. We learn two disjoint patterns of relations, out-relations and in-relations, respectively. Therefore, Equation 1 can again be written as follows.

$$c_{ijk} = \mathbf{W_1}\Phi(\vec{h}_i, \vec{r}_j, \vec{t}_k) \tag{2}$$

$$c_{kji} = \mathbf{W_2}\Phi(\vec{t}_k, \vec{r}_j, \vec{h}_i) \tag{3}$$

where $\mathbf{W_1}$ and $\mathbf{W_2}$ are direction-specific linear transformation matrices. Similar to [16], we learn the importance of each triple denoted by att_{ijk}.

$$att_{ijk} = \text{LeakyReLU}\ (\mathbf{W}_3 c_{ijk}) \tag{4}$$

where $\mathbf{W_3}$ is a linear transformation matrix that is used to calculate attention scores. To get the relative attention values, a softmax function is applied over att_{ijk} as shown in Equation 5.

$$\begin{aligned}\alpha_{ijk} &= \text{softmax}_{jk}\left(att_{ijk}\right) \\ &= \frac{\exp\left(att_{ijk}\right)}{\sum_{n\in\mathcal{N}_i}\sum_{r\in\mathcal{R}_{in}}\exp\left(att_{inr}\right)}\end{aligned} \tag{5}$$

To aggregate information from neighbor u, the feature of node v is updated by:

$$\vec{e}_v = \sigma_1\left(\sum_{u\in\mathcal{N}_v}\sum_{i\in\mathcal{R}_{vu}}\alpha_{viu}c_{viu}\right) \tag{6}$$

As is shown in Figure 2, we incorporate the weighted walk model to get the final entity embedding.

$$\vec{e}_v = \sigma_1\left(\sum_{u\in\mathcal{N}_v}\sum_{i\in\mathcal{R}_{vu}}\alpha_{viu}c_{viu}\right)\|\sigma_1(e_v^{random}) \tag{7}$$

where $\|$ represents channel-wise concatenation, σ_1 is a non-linearity, and e_v^{random} denotes the representation of node v derived from the weighted walk model, which will be explained in next section. After updating entity embedding, the relation embedding is transformed by Equation 8.

$$\vec{r}_j = \mathbf{W}_4 r_j \tag{8}$$

where $\mathbf{W}_4$ is a relation transformation matrix that is used to update relation *j*.

B. Weighted Walk

Unlike Node2Vec [6], which views neighbors as equally important, we think different neighbors in a graph play different roles for a specific node, thus have different contributions to the node's embedding. This is achieved by incorporating attention schemes into the random walk process. We propose an attention-guided random walk aggregation model following the hypothesis that accumulating information from local structural relations of n-step ranges will benefit learning robust embedding.

Therefore, instead of feeding adjacent matrix A to implement random walks, we utilize attention matrix D generated from the graph attention module. As is shown in Equation 9, we define linear combinations of features.

$$H^{i+1} = \widehat{D}^j H^i \tag{9}$$

where H^i represents input entity embedding. The weighted walk algorithm is shown as Algorithm 1. The j^{th} line of H^{i+1} is a representation of entity *j*, denoted as e_j^{random}.

We view the random walking process as a Markov chain. Let D denote the one-step transfer probability matrix (attention

Algorithm 1 The weighted walk algorithm

1: **procedure** RANDOMWALK(D^j, H^i)
2: $H^i \leftarrow WH^i + b$
3: $H^i \leftarrow drop(\sigma(H^i))$
4: $\widehat{D}^j = WeightedPruning(D^j)$
5: **for** $i \leftarrow 1, walklength$ **do**
6: $H^{i+1} = \widehat{D}^j H^i$
7: **end for**
8: $H^{i+1} = \sigma(H^{i+1})$
9: **return** H^{i+1}
10: **end procedure**

matrix), whose state representation space is the set of all entities. D has the following properties:

$$d_{ij} \geq 0, i, j \in \mathcal{E} \tag{10}$$

$$\sum_{j \in \mathcal{E}} d_{ij} = 1, i \in \mathcal{E} \tag{11}$$

Based on that, we have the following definition.

Definition III-B.1. Let the conditional probability be defined as $p_{ij}^{(n)} = P\{X_n = j | X_m = i\}, i, j \in \mathcal{E}$, where entity X_n is n-step neighbor of entity X_m.

Lemma III-B.1. The n-step transfer matrix (attention matrix) has the following property:

$$D^{(n)} = DD^{(n-1)} \tag{12}$$

$$D^{(n)} = D^n \tag{13}$$

From the above properties, we can accumulate n-step information by the one-step transfer matrix, i.e., attention matrix. Line 4 in Algorithm 1 tends to select entities of higher importance, that is, we select a local subgraph. For example, as shown in Figure 1, starting from *Cristiano Ronaldo*, the weighted walk process will choose *Football Player* with higher probability over other neighbors. The final subgraph we obtain will be an n-step local structure composed of relatively important entities.

IV. EXPERIMENTS

We first introduce the experiment settings, including the datasets and experiment descriptions (Section IV-A), and the configurations (Section IV-B), then report the overall performance results on KG completion task (Section IV-C), and finally investigate the contributions of different components of ATTWALK by conducting an ablation study (Section IV-D).

A. Datasets

We evaluate our approach on three publicly available benchmark datasets: WN18RR [13], FB15k-237 [26] and Kinship [27]. We use the standard training, validation, and test sets. FB15k-237 contains entities and relations from Freebase, which is a large common-sense knowledge base. FB15k-237 removes duplicate and inverse relationships to prevent direct prediction. WN18RR is derived from WordNet, a lexical database of semantic relations between words. Similar to FB15k-237, WN18RR also removes duplicates and reverse relationships. The Kinship database consists if relationships of 24 unique entities in two families.

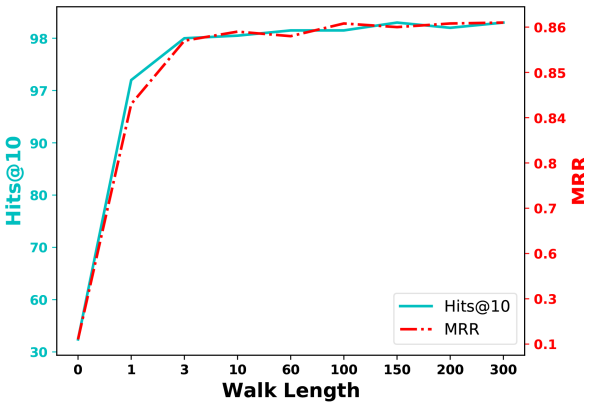


Fig. 3. Impact of walk length on Kinship

B. Configurations

We implement our approach with Pytorch and use Adam to optimize the parameters with an initial learning rate set as 0.001. We run our model under Ubuntu 18.04 on an i9-9900K CPU, equipped with RTX 2080ti 12GB. The embedding size V is set to 200, and the number of negative samples is fixed as 1000. The dropout rate is selected from {0.1, 0.2, 0.3, 0.4}. For algorithm 1, the walk length is tuned amongst {1, 3, 10, 60, 100}. The kernel size of convolution is set as 7×7. We assign label 1 to valid triples and label 0 to negative triples to distinguish them. We use the `CrossEntropyLoss` function as our loss function. Each experiment runs five times and the average number is reported.

C. Results and Analysis

Table I and Table II show the comparison results on all data sets. We can observe that our proposed approach ATTWALK has comparable performance with SOTA baselines on most of the metrics, validating the effectiveness of exploiting both structural and semantic features of a KG. For Kinship, ATTWALK is always the best performer, which outperforms the best baseline by 3.8% on MRR, 2.5% on MR, 6% on Hits@1, 2.3% on Hits@3 and 0.2% on Hits@10 of Kinship, as shown in Table II.

D. Ablation Study

To analyze the effectiveness of each key module in our proposed approach, we investigate an ablation study, which is shown in Table III. In addition, we compare the behavior of our proposed approach when replacing the weighted walk module with NODE2VEC under different aggregation mechanisms and show the results in Table IV. Finally, we compare the effects of different walk lengths, as shown in Figure 3.

1) Effects of Different Modules: As shown in Table III, removing the weighted walk module clearly degrades all the performance metrics, which denotes the effectiveness of weighted walk. When we do not use the updated entity

TABLE I

EXPERIMENTS RESULTS FOR THE LINK PREDICTION TASK ON WN18RR AND FB15K-237 TEST SETS. HITS@N VALUES ARE IN PERCENTAGE. THE BEST SCORE IS IN **BOLD** AND THE SECOND IS UNDERLINED. THE RESULTS OF ALL THE BASELINE METHODS ARE TAKEN FROM THE PREVIOUS PAPERS('-' DENOTES MISSING VALUES).

	WN18RR					**FB15k-237**				
			Hits@N					Hits@N		
	MRR	MR	@1	@3	@10	MRR	MR	@1	@3	@10
TransE [10]	0.226	3384	-	-	50.1	0.294	357	-	-	46.5
DistMult [20]	0.43	5110	39	44	49	0.241	254	15.5	26.3	41.9
ComplEX [22]	0.44	5261	41	46	51	0.247	339	15.8	27.5	42.8
RGCN [15]	-	-	-	-	-	0.248	-	0.151	-	41.7
ConvE [13]	0.43	4187	40	44	52	0.325	244	23.7	35.6	50.1
ConvKB [14]	0.249	3324	5.7	41.7	52.4	0.243	311	15.5	37.1	42.1
KBGAT [16]	0.412	**1921**	-	-	55.4	0.157	270	-	-	33.1
SACN [23]	0.47	-	43	48	54	0.35	-	26	39	54
RotatE [24]	0.476	3340	42.8	49.2	**57.1**	0.338	177	24.1	37.5	53.3
ConvR [25]	0.475	-	44.3	48.9	53.7	0.35	-	26.1	38.5	52.8
CompGCN [18]	0.479	3533	44.3	48.9	53.7	0.355	197	26.4	39.0	53.5
RelWalk [7]	0.451	3232	42	47	51	0.329	**105**	24.3	35.4	50.2
ATTWALK (ours)	**0.483**	2810	**44.5**	**49.7**	56	**0.36**	195.8	**26.8**	**40**	**54.5**

TABLE II

EXPERIMENTS RESULTS FOR THE LINK PREDICTION TASK ON KINSHIP TEST SETS. HITS@N VALUES ARE IN PERCENTAGE. THE BEST SCORE IS IN **BOLD** AND THE SECOND IS UNDERLINED. THE COMPARISONS ARE FROM [16]. WE REPRODUCE THE RESULTS OF KBGAT, RELWALK AND COMPGCN USING [28], [7] AND [18] RESPECTIVELY.

	Kinship				
			Hits@N		
	MRR	MR	@1	@3	@10
TransE [10]	0.309	6.80	0.9	64.3	84.1
DistMult [20]	0.516	5.26	36.7	58.1	86.7
ComplEX [22]	0.823	2.48	73.3	89.9	97.1
RGCN [15]	0.109	25.92	0.3	8.8	23.9
ConvE [13]	0.833	2.00	73.8	91.7	98.1
ConvKB [14]	0.614	3.3	43.62	75.5	95.3
KBGAT [16]	0.548	4.25	36.8	66.5	91.5
CompGCN [18]	0.835	2.06	74.5	91.4	98.3
RelWalk [7]	0.377	4.7	18.4	43.1	92
ATTWALK (ours)	**0.867**	**1.95**	**79.0**	**93.5**	**98.5**

TABLE III

THE EFFECT OF EACH MODULE ON MODEL PERFORMANCE. ATTWALK-W REPRESENTS ATTWALK WITHOUT WEIGHTED WALK MODULE. ATTWALK-O REPRESENTS THERE IS ONLY WEIGHTED WALK MODULE.

	MRR	MR	@1	@3	@10
ATTWALK	**0.483**	**2810**	**44.5**	**49.7**	**56.0**
ATTWALK-W	0.476	3207.4	44.4	49.0	55.0
ATTWALK-O	0.44	4396	40	45.4	52.0

embedding of the graph attention network, but only consider the entity representation obtained from the weighted walk model, we find that experimental results are comparable to ConvE [13], DistMult [20] and RelWalk [7], resulting from that local graph structure features are captured, denoting the effectiveness of Algorithm 1.

2) ATTWALK *v.s. Word Embedding Model:* From Table IV, we find that combining the graph attention network and the weighted walk model as our encoder provides competitive performance for the ConvE [13] score function. Analyzing the experimental results, TransE [10] does not perform as well as DistMult [20] and ConvE [13] after the introduction of local structural features, as TransE [10] tends to express the relationships between individual triples, but on the contrary, ConvE [13] has a stronger expression for capturing the relationships between entities and the local structure of entities. In addition, we evaluate ATTWALK by replacing the weighted walk process with word embedding methods and observe a performance decrease, as shown in Table IV.

3) Effects of Walk Lengths: Finally, we evaluate the effects of different walk lengths. As shown in Figure 3, we find that the performance observes an obvious improvement with the increase of walk lengths l at the beginning, but gets stabilized gradually from around $l = 10$. This is in line with our intuition that the more distant an entity is, the smaller its influence is. Besides, note that the training time increases as the walk length increases, so we need to consider performance improvement and time overhead altogether and make a balance.

V. CONCLUSION

In this paper, we propose a new perspective combining graph structures and semantic information in a KG completion

TABLE IV

LINK PREDICTION PERFORMANCE ON KINSHIP DATASET. X+NODE2VEC (Y) INDICATES THAT WE REPLACE THE ATTENTION-GUIDED WALK MODULE, AS SHOWN IN FIGURE 2, WITH THE CLASSICAL WORD EMBEDDING MODEL NODE2VEC, WHERE X IS THE DECODER FUNCTION AND Y IS THE AGGREGATION METHOD.

Decoder→	TransE			DistMult			ConvE		
Methods↓	MRR	MR	@10	MRR	MR	@10	MRR	MR	@10
X+ATTWALK (sub)	0.068	**38.2**	**16.5**	**0.672**	**3.52**	**92.0**	0.669	14.9	78.8
X+ATTWALK (mult)	**0.070**	47.5	11.1	0.612	3.99	90.6	0.841	2.15	97.8
X+ATTWALK (cross)	0.065	47.9	12.9	0.627	3.87	91.0	**0.867**	**1.95**	**98.5**
X+NODE2VEC (sub)	0.06	47.2	11.9	0.587	4.49	89.2	0.385	27.9	50.0
X+NODE2VEC (mult)	0.055	48.3	10.5	0.557	4.68	88.5	0.816	3.35	92.8
X+NODE2VEC (cross)	0.051	48.3	9.5	0.391	7.50	73.8	0.837	2.79	95.8

task. The idea is simple but effective, and can be combined with many existing KGE approaches. Experiments indicate the effectiveness and provide additional insights that the structural expressiveness of random walks can improve the performance of KGE as well as how to set a walk length. We hope our work can shed some light on and inspire more KGE approaches.

ACKNOWLEDGMENT

This work is supported by grants from the National Natural Science Foundation of China (No.11901578 and No.62102444).

REFERENCES

[1] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

[2] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[3] Yi Ma, Paul A Crook, Ruhi Sarikaya, and Eric Fosler-Lussier. Knowledge graph inference for spoken dialog systems. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5346–5350. IEEE, 2015.

[4] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *EMNLP*, 2015.

[5] Shaoxiong Ji, Shirui Pan, E. Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE transactions on neural networks and learning systems*, PP, 2021.

[6] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[7] Danushka Bollegala, Huda Hakami, Yuichi Yoshida, and Ken-ichi Kawarabayashi. Relwalk a latent variable model approach to knowledge graph embedding. *arXiv preprint arXiv:2101.10070*, 2021.

[8] Binling Nie and Shouqian Sun. Knowledge graph embedding via reasoning over entities, relations, and text. *Future Generation Computer Systems*, 91:426–433, 2019.

[9] Pnina Shachaf. Social reference: Toward a unifying theory. *Library & Information Science Research*, 32(1):66–76, 2010.

[10] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.

[12] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[14] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.

[15] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[16] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019.

[17] Yu Zhao, Huali Feng, Han Zhou, Yanruo Yang, Xingyan Chen, Ruobing Xie, Fuzhen Zhuang, and Qing Li. Eigat: Incorporating global information in local attention for knowledge representation learning. *Knowledge-Based Systems*, page 107909, 2021.

[18] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[20] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[21] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 96–104, 2019.

[22] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[23] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067, 2019.

[24] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[25] Xiaotian Jiang, Quan Wang, and Bin Wang. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 978–987, 2019.

[26] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[27] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, 2018.

[28] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*, 2019.

Dual Contrastive Learning for Unsupervised Knowledge Selection

Xiang Cao[1], Lei Wang[2] and Yujiu Yang[1]

[1]Tsinghua Shenzhen International Graduate School, Shenzhen, China
[2]Ping An Technology (Shenzhen) Co., Ltd., Shenzhen, China
cao-x19@mails.tsinghua.edu.cn, wangleis25@pingan.com.cn,
yang.yujiu@sz.tsinghua.edu.cn

Abstract

Although dialogue systems based on the Seq2Seq model have achieved success, they suffer from tending to generate general responses. Recent works have shown that selecting external knowledge is helpful for dialogue systems to generate informative and diverse responses. However, selecting appropriate knowledge from an unlabeled knowledge set, which is referred to as unsupervised knowledge selection, remains a tricky challenge. Therefore, we propose a dual contrastive method, which utilizes two source-target pairs which are based on the same knowledge set to construct dual contrasts. Specifically, for a source utterance, we consider its paired and unpaired target response as a positive and negative sample, then obtain the positive and negative posterior distribution over the knowledge candidates set, respectively. Then we lead the prior distribution to be close to the positive posterior distribution and distant from the negative one. Similarly, the posterior distribution is treated with the same criterion. Experimental results show that our method improves generated responses in terms of BLUE, DISTINCT, and knowledge utilization. Our codes are available at https://github.com/CaoXiang1997/DualCL4UKS.

1. Introduction

A dialogue system aims to produce an appropriate response given a post as its input. Recently, the generative method based on the encoder-decoder frame [4] has attracted considerable attention for generating fluent responses [15, 17]. However, it tends to generate general and boring responses like "I don't know", lending conversations into unattractive and boring situations.

Existing works have shown that leveraging external knowledge is helpful for dialogue systems to improve informativeness and diversity of generated responses [6, 10, 14]. External knowledge is sometimes in the form of multiple utterances in a set of knowledge [19, 5]. Intuitively, external knowledge builds an information bridge for dialogue systems from the source to the target. However, not all utterances in the knowledge set help models generate appropriate responses, and manual annotations are often expensive. Therefore, it is necessary to select appropriate knowledge from an unlabeled knowledge set, referred to as unsupervised knowledge selection.

For unsupervised knowledge selection, a tricky challenge is the lack of supervisory signals. Several works proposed attentive methods which use attention mechanism [10] or its more powerful varieties [14] to calculate the probability distribution over the knowledge set and select knowledge softly. However, those attentive methods only relied on the distant supervisory signal from the cross-entropy loss between the generated response and the target response to supervise knowledge selection, helpless to the lack of the supervisory signal. [10] attempted to use target response to compute posterior distribution as guidance for prior distribution, but still didn't provide a strong enough supervisory signal.

In this paper, inspired by previous works on contrastive learning [2, 7], we propose a dual contrastive method for unsupervised knowledge selection. From contrastive learning, the model benefits from the contrast between positive samples and negative samples. We think that appropriately selected knowledge is helpful for the model to distinguish positive samples from negative ones. Specifically, for a

DOI reference number: 10.18293/SEKE2022-054

source utterance, we consider its paired target response as the positive response and an unpaired target response as the negative response, compute positive and negative posterior distribution on the positive and negative response, respectively, and lead the prior distribution to be close to the positive posterior distribution and distant from the negative one. Similarly, we utilize the same criterion for the posterior distribution. Our contributions are listed as follows.

- As far as we know, we are the first to introduce contrastive learning into unsupervised knowledge selection, which provides an alternative solution for the label unavailability issue.
- We propose a dual contrastive method for unsupervised knowledge selection, which learns prior/posterior distribution from contrasts between positive and negative posterior/prior distributions.
- Experimental results show that our method outperforms other existing competitive methods on diversity and knowledge incorporation of generated responses and almost flats on other metrics.

2. Related Work

Sequence to sequence (Seq2Seq) models [4] promote the development of dialogue systems, but it suffers from generating general responses. Recently, some works utilize external knowledge to help dialogue systems generate diverse and informative responses. In some early works [6, 12], the model encoded external text entirely into a vector, which led to irrelevant knowledge noise in generated responses. Therefore, unsupervised knowledge selection became a research hotspot whereas the lack of labels. [10] proposed a prior-posterior framework that computes posterior distribution by the ground-truth response and drives the prior distribution to approach the posterior distribution. [14] proposed a global-to-local knowledge selection mechanism where the global knowledge selection module forms a topic transition vector and the local knowledge selection module select knowledge at each decoding step under the guidance of the topic transition vector. [16] proposed a teacher-student framework where the teacher builds response-aware document memory given the ground-truth response and the student learns response-anticipated document memory from the teacher. [3] introduced knowledge distillation to address the exposure bias issue of knowledge selection. [11] proposed recurrent knowledge interaction among decoding steps and introduced a knowledge copy mechanism to copy words from external knowledge. [8] proposed a sequential latent model which uses sequential latent to model the knowledge selection process in multi-turn dialogue generation.

However, existing works use one single source-target pair to select knowledge, ignoring the difference between knowledge selected by two source-target pairs. That results in more attention to the common information than the special information of knowledge candidates, which is bad for the model to incorporate relevant knowledge and generate diverse responses. In this paper, We propose a dual contrastive method for unsupervised knowledge selection, which leads the model to learn the difference in knowledge selection between two source-target pairs and generate responses based on specially selected knowledge.

3. Model

As presented in Figure 1, the architecture overview of our model is generally based on a sequence-to-sequence frame. We let $\left\{(x^i, y^i)\right\}_{i=1}^{n}$ denote a set of multi-turn dialogue, where n is the turn number, and x^i and y^i are the source utterance and the target utterance of the i-th turn, respectively. $K = \left\{k^j\right\}_{j=1}^{N}$ denotes a set of knowledge utterances, where N is the number of utterances in K, and k^j is the j-th knowledge utterance. Our model's goal is to select correct a knowledge utterance k^j from K and generate appropriate responses y^i for each x^i.

3.1. Base Architecture

Our method is based on an encoder-decoder architecture with a prior-posterior knowledge selector which uses the posterior distribution to guide the prior distribution.

Encoder We implement a source encoder with a bidirectional gated recurrent unit (GRU), which encode source utterance x^i into a forward hidden state $\overrightarrow{h}_x^i$ and a backward hidden state $\overleftarrow{h}_x^i$ for each x^i. We concatenate the last hidden states in two directions into h_x^i as the representation vector of x^i as follows.

$$\overrightarrow{h}_t = GRU(x_t^i, \overrightarrow{h}_{t-1}) \tag{1}$$

$$\overleftarrow{h}_t = GRU(x_t^i, \overleftarrow{h}_{t+1}) \tag{2}$$

$$h_x^i = [\overrightarrow{h}^i_{|x^i|}; \overleftarrow{h}^i_1] \tag{3}$$

where $[;]$ represents vector concatenation, $|x^i|$ represents the token number of x^i. In this way, we encode x^i as h_x^i for each i.

We implement a knowledge encoder with the same structure as the source encoder, but they don't share any parameters. Similarly, we concatenate the last hidden states of two directions into an overall vector h_k^j for each k^j. Moreover, we use the knowledge encoder to encode y^i as h_y^i for each i.

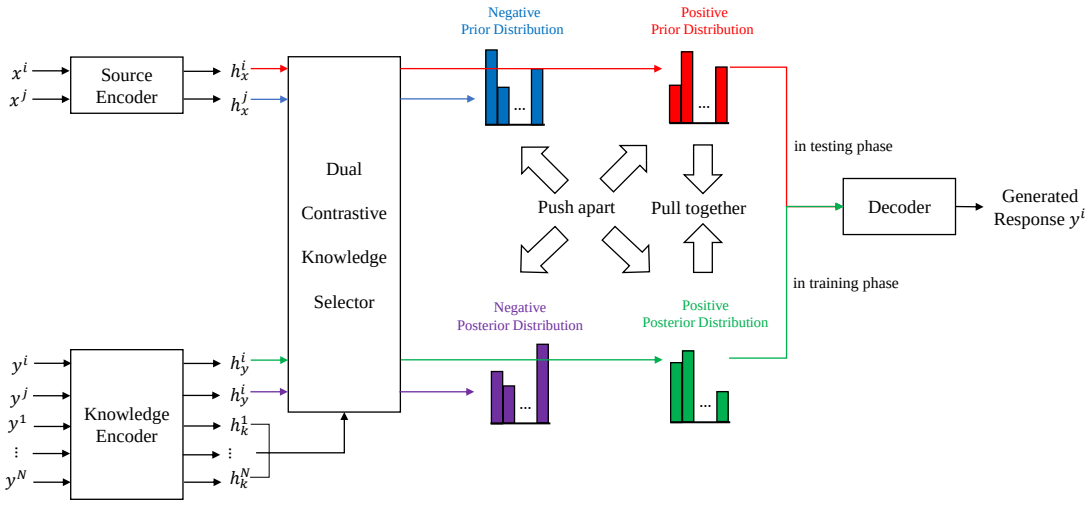


Figure 1. The architecture overview of our model. In this figure, we only show contrasts where the i-th turn (x^i, y^i) is the positive sample and the j-th turn (x^j, y^j) is the negative one.

Knowledge Selector The goal of the knowledge selector is to select appropriate knowledge from the knowledge set. Inspired by [10], we set two modes for the knowledge selector – the prior and posterior modes. In the prior mode, the knowledge selector computes a prior distribution p_x^i using the source utterance x^i. In the posterior mode, the knowledge selector computes a posterior distribution p_x^i using the target utterance y^i. The knowledge selector is implemented with attention mechanism [1] as follows.

$$p(k|z) = softmax(h_z \cdot [h_k^1, \cdots, h_k^n]) \tag{4}$$

where $z \in \{x^i\}_{i=1}^n \cup \{y^i\}_{i=1}^n$ is the representation vector of an source or target utterance. For convenience, we use p_x^i and p_y^i as simplifications of $p(k|x^i)$ and $p(k|y^i)$, respectively.

In the training phase, we use the posterior distribution as knowledge selection distribution to sample a knowledge utterance. In the testing phase, we have no choice but to use the prior distribution, because the target utterance is to be generated and can not be used to compute the posterior distribution.

To ensure the knowledge selector work in the testing phase, we use the posterior distribution to guide the prior distribution in the training phase. Therefore, we introduce the Kullback-Leibler Divergence (KLD) loss to minimize the distance between the prior distribution and the posterior distribution.

$$\ell_{KLD}^i = KLD(p_y^i || p_x^i) = p_y^i \log \frac{p_y^i}{p_x^i} \tag{5}$$

When the knowledge selection distribution $p(k)$is given, the selected knowledge utterance $sk \sim p(k)$ is sampled according to it.

Decoder The decoder integrates the selected knowledge h_{sk}^i and generates response word by word. We use a hierarchical gated fusion unit (HGFU) [18] to implement it. An HGFU consists of two GRUs, which are fed by the word generated in the last step y_{t-1} and the selected knowledge h_K, respectively, as follows.

$$\begin{aligned} s_{y,t}^i &= GRU(emb(y_{y,t-1}^i), s_{t-1}^i, c_t^i) \\ s_{k,t}^i &= GRU(h_{sk}^i, s_{t-1}^i, c_t^i) \end{aligned} \tag{6}$$

where emb is the embedding layer, s_{t-1}^i is the last hidden state of the decoder, c_t^i is the attentive context vector.

Then, HGFU combines the s_t^y and s_t^k with a soft gate g as follows.

$$s_t^i = g\dot{s}_{y,t}^i + (1-g) \odot s_{k,t}^i \tag{7}$$

g is computed by s_t^y and s_t^k through multilayer perceptrons and control the their contributions to the final hidden state s_t.

The word is sampled from a distribution computed by s_t^i and c_t^i as follows.

$$y_t^i \sim p_t^i = softmax(W_o[s_t^i; c_t^i]) \tag{8}$$

where W_o is the parameters of the output layer.

Loss Function We introduce the negative-log likelihood(NLL) loss to measure the difference between the response generated by the model and the target response as follows.

$$\ell^i_{NLL} = -\frac{1}{|y^i|}\sum_{t=1}^{|y^i|}\log p(y^i_t|y^i_{t-1}, x^i, sk^i) \tag{9}$$

Like [10], we introduce the bag-of-words(BOW) loss to ensure the accuracy of the selected knowledge as follows.

$$\ell^i_{BOW} = -\frac{1}{m}\sum_{t=1}^{m}\log p(y_t|k) \tag{10}$$

Therefore, the total loss function of our model is as follows.

$$\ell = \ell_{KS} + \ell_{NLL} + \ell_{BOW} \tag{11}$$

3.2. Dual Contrastive Knowledge Selector

We think that an appropriately selected knowledge is helpful not only for approaching the target response but also for distinguishing the target response from other responses. Therefore, we propose two kinds of contrastive loss as follows. For the posterior distribution p^i_y, we use the prior distribution p^i_x of the same turn as the positive and the prior distributions $\{p^j_x\}^n_{j=1,j\neq i}$ of other turns as the negatives. Then, we minimize the distance of p^i_y from the positive prior distribution p^i_x and maximize the average distance of p^i_y from the negative prior distributions $\{p^j_x\}^n_{j=1,j\neq i}$. In this way, We introduce the prior contrast and propose the prior contrastive loss as follows.

$$\begin{aligned}\ell^i_{PRIOR} &= p^i_y \log\frac{p^i_y}{p(k|x^i)} - \frac{1}{n-1}\sum_{j=1,j\neq i}^{n} p^i_y \log\frac{p^i_y}{p^j_x} \\ &= -p^i_y(\log p^i_x - \frac{1}{n-1}\sum_{j=1,j\neq i}^{n}\log p^j_x)\end{aligned} \tag{12}$$

To ensure ℓ_{PRIOR} is positive, we change it to its final form as follows.

$$\ell^i_{PRIOR} = -p^i_y[\log p^i_x + \frac{1}{n-1}\sum_{j=1,j\neq i}^{n}\log(1-p^j_x)] \tag{13}$$

Similarly, for the prior distribution p^i_x, we use the posterior distribution p^i_y of the same turn as the positive and the posterior distributions $\{p^j_y\}^n_{j=1,j\neq i}$ of other turns as the negatives. Then, we minimize the distance of p^i_x from the positive posterior distribution p^i_y and maximize the average distance of p^i_y from the negative posterior distributions $\{p^j_y\}^n_{j=1,j\neq i}$. In this way, we introduce the posterior contrast and propose the posterior contrastive loss as follows.

$$\begin{aligned}\ell^i_{POST} &= p^i_y \log\frac{p^i_y}{p(k|x^i} - \frac{1}{n-1}\sum_{j=1,j\neq i}^{n} p^j_y \log\frac{p^j_y}{p^i_x} \\ &= [p^i_y \log p^i_y - \frac{1}{n-1}\sum_{j=1,j\neq i} p^j_y \log p^j_y] \\ &\quad - [p^i_y \log p^i_x - \frac{1}{n-1}\sum_{j=1,j\neq i} p^j_y \log p^i_x]\end{aligned} \tag{14}$$

For the whole multi-turn dialogue, we have

$$\sum_{i=1}^{n}[p^i_y \log p^i_y - \frac{1}{n-1}\sum_{j=1,j\neq i} p^j_y \log p^j_y] = 0 \tag{15}$$

Therefore, ℓ^i_{POST} can be simplified as follows.

$$\ell^i_{POST} = -p^i_y \log p^i_x + \frac{1}{n-1}\sum_{j=1,j\neq i} p^j_y \log p^i_x \tag{16}$$

Similarly, we change ℓ^i_{POST} to ensure it is positive as follows.

$$\ell^i_{POST} = -p^i_y \log p^i_x - \frac{1}{n-1}\sum_{j=1,j\neq i} p^j_y \log(1-p^i_x) \tag{17}$$

Overall, the total loss function of knowledge selection is as follows.

$$\ell^i_{KS} = \ell^i_{KL} + \alpha\cdot\ell^i_{PRIOR} + \beta\cdot\ell^i_{POST} \tag{18}$$

where α and β are coefficients to control the contribution of ℓ^i_{PRIOR} and ℓ^i_{POST}, respectively.

4. Experiments

4.1. Experiment Settings

Datasets We carry experiments on two open-domain knowledge-grounded dialogue datasets, namely PersonaChat [19] and Wizard-of-Wikipedia [5]. Although Wizard-of-Wikipedia has labels for knowledge selection, we did not use them because we focus on improvements in unsupervised knowledge selection.

Baselines We compared our models with the following baselines.

Table 1. Automatic Evaluation on PersonaChat and Wizard-of-Wikipedia.

Datasets	Models	BLEU-1 / 2 / 3	Distinct-1 / 2 / 3	Knowledge-R / P / F1
PersonaChat	Seq2Seq	0.1764 / 0.0725 / 0.0315	0.0136 / **0.1015** / **0.2908**	0.0062 / 0.0206 / 0.0095
	PostKS	0.1736 / 0.0720 / 0.0326	0.0136 / 0.0968 / 0.2676	0.0098 / 0.0407 / 0.0158
	our model	**0.1799** / **0.0739** / **0.0333**	**0.0144** / 0.1011 / 0.2804	**0.0110** / **0.0430** / **0.0176**
Wizard-of-Wikipedia (Test Seen)	Seq2Seq	0.1802 / 0.0608 / 0.0248	0.0480 / 0.2575 / 0.5468	0.0175 / 0.2427 / 0.0327
	PostKS	0.1936 / 0.0679 / 0.0278	0.0487 / 0.2642 / 0.5539	0.0245 / **0.3381** / 0.0457
	our model	**0.1961** / **0.0686** / **0.0281**	**0.0500** / **0.2759** / **0.5709**	**0.0248** / 0.3255 / **0.0461**
Wizard-of-Wikipedia (Test Unseen)	Seq2Seq	0.1735 / 0.0560 / 0.0227	**0.0397** / **0.2148** / 0.4849	0.0143 / 0.1827 / 0.0264
	PostKS	0.1805 / 0.0575 / 0.0221	0.0325 / 0.2077 / **0.4951**	0.0208 / 0.2557 / 0.0384
	our model	**0.1808** / **0.0585** / **0.0228**	0.0318 / 0.2001 / 0.4805	**0.0210** / **0.2620** / **0.0388**

Table 2. The ablation results on the PersonaChat dataset of our model.

Models	BLEU-1 / 2 / 3	Distinct-1 / 2 / 3	Knowledge-R / P / F1
our model	**0.1799** / **0.0739** / **0.0333**	0.0144 / **0.1011** / **0.2804**	**0.0110** / 0.0430 / **0.0176**
w/o prior contrast	0.1745 / 0.0725 / 0.0330	**0.0146** / 0.0995 / 0.2748	0.0102 / 0.0414 / 0.0163
w/o posterior contrast	0.1785 / 0.0737 / 0.0333	0.0144 / 0.0991 / 0.2739	0.0109 / **0.0437** / 0.0175

- **Seq2Seq** [4] is an attentive seq2seq model that does not have access to external knowledge.
- **PostKS** [10] is an attentive seq2seq that selects knowledge with the posterior distribution in the training phase and the prior distribution in the testing phase.

Implementation. We use a bidirectional GRU with 400 hidden states for each layer as our encoder and 1-layer GRUs with 800 hidden states in our decoder. All encoders and decoders do not share any parameters. We set the word embedding size to be 300 and initialized it using GloVe [13]. We use a vocabulary table that has no more than 20,000 words. We use an Adam optimizer [9], where the batch size of 16, and the learning rate is 5e-4. In the first five epochs, we minimize the BOW loss only for pre-training the knowledge selector. In the remaining epochs, we minimize the sum of all losses. We evaluated our model on the validation set every 100 steps and stopped training when the model did not update the minimal loss for a whole epoch.

Evaluation. We adopted several automatic metrics to perform the evaluation. *BLEU-1/2* and *Distinct-1/2* are two widely used metrics for evaluating the quality and diversity of generated responses. Due to the lack of labels, the quality of selected knowledge is hard to be measured directly. Following [10], we use *Kownledge-R/P/F1* to evaluate the knowledge quality of generated responses via measuring the relevancy between generated responses and the knowledge set. Specifically, given the set of non-stop words in a response Y and in the knowledge set K, denoted by W_Y and W_K, Knowledge-R/P/F1, denoted by $R/P/F1$ respectively, are defined as follows.

$$R = \frac{|W_Y \bigcap W_K|}{|W_K|} \tag{19}$$

$$P = \frac{|W_Y \bigcap W_K|}{|W_Y|} \tag{20}$$

$$F1 = 2 \cdot \frac{R \cdot P}{R + P} \tag{21}$$

4.2. Evaluation Results.

Effect of Dual Contrastive Learning The evaluation results are summarized in Table 1. Bold numbers show the best results among all models. We observe that our model outperforms baseline models in terms of knowledge utilization of generated responses on almost all datasets. For example, Knowledge-R/P/F1 on PersonaChat is increased from 0.0098/0.0407/0.0158 (PostKS) to 0.0110/0.0430/0.0176 (our model), indicating the improvement in terms of the quality of knowledge selection.

Ablation Study The ablation results on PersonaChat of our model are reported in Table 2. We observe that both prior and posterior contrast contribute to our model because the performance degrades without any of them. By comparison, the prior contrast contributes more, where we think the reason is that the prior distribution is directly used in the testing phase. The prior contrast directly increases knowledge selection compared to the posterior contrast.

5. Conclusion

This paper proposes a dual contrastive method for unsupervised knowledge selection in dialogue systems, which is the first work that introduces contrastive learning into knowledge selection in dialogue systems. Experiment results show that our model has improved on diversity and knowledge incorporation of generated responses. As for future work, we plan to extend our contrastive method to Transformer-based architecture.

ACKNOWLEDGMENTS

This research was supported in part by the Ministry of Science and Technology of China under Grant No. 2020AAA0104200, and the Shenzhen Key Laboratory of Marine IntelliSense and Computation under Contract ZDSYS20200811142605016.

References

[1] BAHDANAU, D., CHO, K., AND BENGIO, Y. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR* (2015).

[2] CAI, H., CHEN, H., SONG, Y., DING, Z., BAO, Y., YAN, W., AND ZHAO, X. Group-wise contrastive learning for neural dialogue generation. In *Proc. of EMNLP* (2020).

[3] CHEN, X., MENG, F., LI, P., CHEN, F., XU, S., XU, B., AND ZHOU, J. Bridging the gap between prior and posterior knowledge selection for knowledge-grounded dialogue generation. In *Proc. of EMNLP* (2020).

[4] CHO, K., VAN MERRIENBOER, B., BAHDANAU, D., AND BENGIO, Y. On the properties of neural machine translation encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014* (2014).

[5] DINAN, E., ROLLER, S., SHUSTER, K., FAN, A., AULI, M., AND WESTON, J. Wizard of wikipedia knowledge-powered conversational agents. In *Proc. of ICLR* (2019).

[6] GHAZVININEJAD, M., BROCKETT, C., CHANG, M., DOLAN, B., GAO, J., YIH, W., AND GALLEY, M. A knowledge-grounded neural conversation model. In *Proc. of AAAI* (2018).

[7] GUTMANN, M., AND HYVARINEN, A. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.* (2012).

[8] KIM, B., AHN, J., AND KIM, G. Sequential latent knowledge selection for knowledge-grounded dialogue. In *Proc. of ICLR* (2020).

[9] KINGMA, D. P., AND BA, J. Adam A method for stochastic optimization. In *Proc. of ICLR* (2015).

[10] LIAN, R., XIE, M., WANG, F., PENG, J., AND WU, H. Learning to select knowledge for response generation in dialog systems. In *Proc. of IJCAI* (2019).

[11] LIN, X., JIAN, W., HE, J., WANG, T., AND CHU, W. Generating informative conversational response using recurrent knowledge-interaction and knowledge-copy. In *Proc. of ACL* (2020).

[12] PARTHASARATHI, P., AND PINEAU, J. Extending neural generative conversational model using external knowledge sources. In *Proc. of EMNLP* (2018).

[13] PENNINGTON, J., SOCHER, R., AND MANNING, C. D. Glove global vectors for word representation. In *Proc. of EMNLP* (2014).

[14] REN, P., CHEN, Z., MONZ, C., MA, J., AND DE RIJKE, M. Thinking globally, acting locally distantly supervised global-to-local knowledge selection for background based conversation. In *Proc. of AAAI* (2020).

[15] SHANG, L., LU, Z., AND LI, H. Neural responding machine for short-text conversation. In *Proc. of ACL* (2015).

[16] TIAN, Z., BI, W., LEE, D., XUE, L., SONG, Y., LIU, X., AND ZHANG, N. L. Response-anticipated memory for on-demand knowledge integration in response generation. In *Proc. of ACL* (2020).

[17] VINYALS, O., AND LE, Q. V. A neural conversational model. *CoRR abs1506.05869* (2015).

[18] YAO, L., ZHANG, Y., FENG, Y., ZHAO, D., AND YAN, R. Towards implicit content-introducing for generative short-text conversation systems. In *Proc. of EMNLP* (2017).

[19] ZHANG, S., DINAN, E., URBANEK, J., SZLAM, A., KIELA, D., AND WESTON, J. Personalizing dialogue agents I have a dog, do you have pets too. In *Proc. of ACL* (2018).

Dynamic Heterogeneous Information Network Embedding in Hyperbolic Space

Dingyang Duan*, Daren Zha*, Xiao Yang** and Xiaobo Guo(✉)*

*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
*School of Cyber Security,University of Chinese Academy of Sciences, Beijing, China
**Aerospace Internet of Things Technology Co.,Ltd, Beijing, China
**China Aerospace Times Electronics Co., Ltd, Beijing, China

***Abstract*—Heterogeneous information network (HIN) embedding, aiming to project HIN into a low-dimensional space, has attracted considerable research attention. Existing heterogeneous graph representation learning methods also take temporal evolution into consideration in Euclidean space which, however, underestimates the inherent complex and hierarchical properties in many real-world temporal networks, leading to sub-optimal embeddings. To explore these properties of a dynamic heterogeneous network, we propose a dynamic hyperbolic heterogeneous embedding(DyHHE) model that fully takes advantage of the hyperbolic geometry and structural heterogeneity. More specially, to capture the structure and semantic relations between nodes, we employ the meta-path guided random walk to sample the sequences for each node. Then DyHHE maps the temporal graph into hyperbolic space, and capture the structural heterogeneity and evolving behaviors by facilitating the proximity measurement. Experimental results on two real-world datasets demonstrate the superiority of DyHHE, as it consistently outperforms competing methods in link prediction task.**

***Index Terms*—Dynamic Graphs, Hyperbolic Space, Heterogeneous Information Network**

I. Introduction

Modeling data in the real world as heterogeneous information networks(HINs) can capture the internal relations of rich, complex data across various modalities. However, many real-world graphs are dynamic where graph structures constantly evolve over time. They are usually represented as sequence of graph snapshots at different time steps [1]. Examples include co-authorship networks where authors may periodically switch social network whose users may develop their multiple-type connections(follow, reply, retweet, etc) with others over time. The dynamics of a network and the structural heterogeneity provide abundant information for encoding nodes. So far, a number of HIN embedding methods have been proposed such as metapath2vec [2] and HAN [3]. These methods have overlooked a problem, that is, the formation of the neighbors is actually in order, it is related to time. There has been an ever-increasing amount of research on dynamic networks like DySAT [4] and HDGAN [5]. However, in the vast majority of these works, the space used for representing networks is Euclidean. In recent years, it has been suggested that complex networks may have underlying hyperbolic geometry and that hyperbolic space can better represent the structure of networks [6]. One fundamental property of hyperbolic space is that it expands exponentially and can be regarded as a smooth version of trees, abstracting the hierarchical organization. Despite the recent achievements in hyperbolic graph embedding, attempts on temporal heterogeneous networks are still scant. To fill this gap, in this work, we propose a novel dynamic hyperbolic heterogeneous embedding model, which fully takes advantage of the hyperbolic geometry and structural heterogeneity to capture the spatial dependency and temporal regularities of evolving networks via a recurrent learning paradigm. In summary, the main contributions are stated as follows:

- We propose a novel hyperbolic temporal graph embedding model on heterogeneous network, named DyHHE, to learn temporal regularities and implicitly hierarchical organization.
- We devise a hyperbolic structural network(HSN) module to preserve the HIN structure and semantic correlations in hyperbolic spaces based on the meta-path guided random walk. Then we apply a hyperbolic temporal network(HTN) module to effectively extract the diverse scope of historical information. To the best of our knowledge, this is the first study on dynamic heterogeneous network embedding in hyperbolic space.
- Experimental results on two real-world datasets demonstrate the superiority of DyHHE, as it consistently outperforms competing methods in link prediction task. The ablation study further gives insights into how each proposed component contributes to the success of the model.

II. Related Works

Recently, some methods have been proposed representation learning methods for HIN. Heterogeneous networks are usually characterized by meta-paths to find hidden relationships between nodes. Metapath2vec [2] obtains a corpus through random walks based on meta-paths, and uses skip-Gram for training. HAN [3] applies the attention mechanism to heterogeneous graphs through meta-path based neighbors. Real-world networks are not static, on the contrary, many networks are constantly changing, such as social networks. HDGAN [5] attempts to use the attention mechanism to take the

***This paper was withdrawn by the authors due to errors in the paper.

DOI reference number: 10.18293/SEKE2022-087

heterogeneity and dynamics of the network into account at the same time. DySAT [4] use the scaled dot-product form of attention to learn dynamic graph embedding. Most of the prevalent methods are built-in Euclidean space which, however, may underemphasize the intrinsic power-law distribution and hierarchical structure. Existing representation learning in hyperbolic space such as HGCN [11] and PoincareEmb [21] mainly focus in static graph. Despite the recent achievements in hyperbolic graph embedding, attempts on dynamic heterogeneous networks are still scant, which motivates us to explore hyperbolic geometry on dynamic heterogeneous networks.

III. Preliminaries

In this section, we first present the problem formulation of dynamic heterogeneous graph, then we introduce some fundamentals of hyperbolic geometry, which are essential in our proposed framework.

Definition 1. *Heterogeneous Information Network(HIN) [7].* An HIN is defined as a graph $G = (V, E)$, in which V and E are the sets of nodes and edges. Each node $v \in V$ and each edge $e \in E$ are associated with their mapping functions $\phi(v) : V \to \mathcal{V}$ and $\psi(e) : E \to \mathcal{E}$ respectively. $\mathcal{V}$ and $\mathcal{E}$ denote the sets of node and relation types, where $|\mathcal{V}| + |\mathcal{E}| > 2$.

Definition 2. *Meta-path [2].* Given a HIN $G = (V, E)$, a meta-path $\mathcal{P}$ is a sequence of node types $\mathcal{V}_{v_1}, \mathcal{V}_{v_2}, ..., \mathcal{V}_{v_n}$ connected by edge types $\mathcal{E}_{e_1}, \mathcal{E}_{e_2}, ..., \mathcal{E}_{e_{n-1}}$: $\mathcal{P} = \mathcal{V}_{v_1} \xrightarrow{\mathcal{E}_{e_1}} \mathcal{V}_{v_2} \xrightarrow{\mathcal{E}_{e_2}} ... \xrightarrow{\mathcal{E}_{e_{n-1}}} \mathcal{V}_{v_n}$. A meta-path instance consists of specific nodes and edges, e.g., $a_1 \xrightarrow{write} p_1 \xrightarrow{publish} v_1$.

Definition 3. *Dynamic Heterogeneous Graph.* A heterogeneous temporal is defined as a graph $G =< V, E, A, T >$, from definition 1, we know that $|\mathcal{V}| + |\mathcal{E}| > 2$, where V represents the node type and E represents the edge type. A represents an event that sequence formed for each node's neighbors, and T is a time stamp. By definition 2, we can get the neighbor set of node i in the heterogeneous network. The neighbor formation sequence of node i refers to organizing the neighbors of the nodes in the network as a sequence based on the time of neighbor interaction events [8].

Then, we introduce some concepts of geometry to make this article more clear. A Riemannian manifold $\mathcal{M}$ is a space that generalizes the notion of a 2D surface to higher dimensions [9]. For each point $\mathbf{x} \in \mathcal{M}$, it associates with a **tangent space(Euclidean)** $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ of the same dimensionality as $\mathcal{M}$. Intuitively, $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ contains all possible directions in which one can pass through $\mathbf{x}$ tangentially (see Fig. 1). There are multiple models that can be used to represent hyperbolic space, each having different advantages. The Poincaré ball model is the best model for low dimensional visualizations of the embeddings. The Poincaré ball model with negative curvature $-c(c \geq 0)$ corresponds to the Riemannian manifold $(\mathbb{H}^{n,c}, g_{\mathbb{H}})$, where $\mathbb{H}^{n,c} = \{\mathbf{x} \in \mathbb{R}^n : c||\mathbf{x}||^2 \leq 1\}$ is an open n-dimensional ball. if $c = 0$, it degrades to Euclidean space, i.e., $\mathbb{H}^{n,c} = \mathbb{R}^n$. In addition, [9] shows how Euclidean and hyperbolic spaces can be continuously deformed into each other and provide a principled manner for basic operations (e.g., addition and multiplication) as well as essential functions (e.g., linear maps and softmax layer) in the context of neural networks and deep learning.

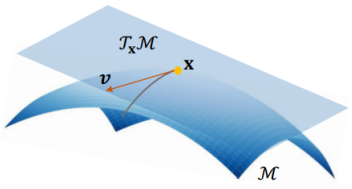


Fig. 1. The tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ and a tangent vector v, along the given point $\mathbf{x}$ of a curve traveling through the manifold $\mathcal{M}$.

IV. Proposed Model

The overall framework of the proposed model DyHHE is illustrated in Fig. 2. DyHHE has two primary modules: hyperbolic structural module and hyperbolic temporal module, which benefits from the expressiveness of both hyperbolic embeddings and temporal evolutionary embeddings. As sketched in Fig. 2, DyHHE is a recurrent learning paradigm and falls into the prevalent discrete-time temporal graph architecture formulated by (1). More specifically, DyHHE can be summarized as two procedures: (a)Given the original input node feature, this procedure projects it into hyperbolic space, and preserve the structure by facilitating the proximity between the node $v \in V$ and its neighborhoods $c_{\mathcal{V}} \in C_{\mathcal{V}}(v)$ with type $\mathcal{V}$. We use meta-path guided random walks [2] to obtain heterogeneous neighborhoods of a node. (b)These sequences of node representations then feeds as input to the temporal recurrent module to capture the sequential patterns. Furthermore, we propose an attention mechanism based on the hyperbolic proximity to obtain the attentive hidden state. Owning to the superiorities of self-attention, this unit attending on multiple historical latent states to get a more informative hidden state. We elaborate on the details of each respective module in the following paragraphs.

$$H_t(\phi) = f_2(f_1(A_t, X_t), H_{t-1}) \tag{1}$$

A. Feature Map

Before going into the details of each module, we first introduce two bijection operations, the exponential map and the logarithmic map, for mapping between hyperbolic space and tangent space with a local reference point [10], [11], as presented below.

In this work, we use Poincaré model with constant curvature $c = 1$ as the hyperbolic space for entity embeddings [12]. In particular, we denote d-dimensional Poincaré centered at origin as $\mathbb{H}^{n,c} = \{\mathbf{x} \in \mathbb{R}^n : c||\mathbf{x}||^2 \leq 1\}$, where $||\cdot||$ is the Euclidean norm. The Poincaré model of hyperbolic space is equipped with Riemannian metric:

$$g_x^{\mathbb{H}} = \lambda_{\mathbf{x}'}^2 g^{\mathbb{R}} \tag{2}$$

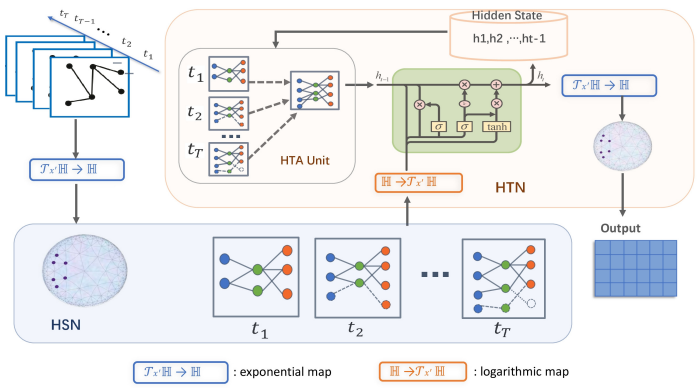


Fig. 2. Architecture of DyHHE.

where $\lambda^c_{\mathbf{x}'} := \frac{2}{1-c||x'||^2}$ and $g^{\mathbb{R}}$ denotes the Euclidean metric, i.e., $g^{\mathbb{R}} = \mathbb{I}$. The mobius addition $\oplus$ defined on Poincaré model with curvature c is given by:

$$u \oplus v := \frac{(1 + 2c\langle u, v\rangle + c||v||^2)u + (1 - c||u||^2)v}{1 + 2c\langle u, v\rangle + c^2||u||^2||v||^2}. \quad (3)$$

For each point $\mathbf{x}' \in \mathbb{H}^{d,c}$, the tangent space $\mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ is the Euclidean vector space containing all tangent vectors at $\mathbf{x}'$. For $\mathbf{x}' \in \mathbb{H}^{d,c}$, $a \in \mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$, $b \in \mathbb{H}^{d,c}$, and $a \neq 0$, $b \neq \mathbf{x}'$. One can map vectors in $\mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ to vectors in $\mathbb{H}^{d,c}$ through exponential map $\exp^c_{\mathbf{x}'}(\cdot) : \mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c} \to \mathbb{H}^{d,c}$ as follows:

$$\exp^c_{\mathbf{x}'}(a) = \mathbf{x}' \oplus^c (\tanh(\frac{\sqrt{c}\lambda^c_{\mathbf{x}'}||a||}{2})\frac{a}{\sqrt{c}||a||}) \quad (4)$$

Conversely, the logarithmic map $\log^c_{\mathbf{x}'}(\cdot) : \mathbb{H}^{d,c} \to \mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ maps vectors in $\mathbb{H}^{d,c}$ back to vectors in $\mathcal{T}_{\mathbf{x}'}$, in particular:

$$\log^c_{\mathbf{x}'}(b) := \frac{2}{\sqrt{c}\lambda^c_{\mathbf{x}'}}\operatorname{artanh}(\sqrt{c}||-\mathbf{x}' \oplus^c b||)\frac{-\mathbf{x}' \oplus^c b}{||-\mathbf{x}' \oplus^c b||} \quad (5)$$

Also, the hyperbolic distance between $u, v \in \mathbb{H}^{d,c}$ is:

$$d_c(u, v) = (2\sqrt{c}\operatorname{artanh}(\sqrt{c}||-u \oplus^c v||)) \quad (6)$$

B. Hyperbolic Structural Network(HSN)

On a dynamic heterogeneous graph, various kinds of interactions are constantly being established over time, which can be regarded as a series of observed heterogeneous events. We aim to learn the representation of nodes to preserve the structure and semantic correlations in hyperbolic spaces for each snapshot. In each time step, HSN is employed to preserve the structure by facilitating the proximity between the node $v \in V$ and its neighborhoods $c_{\mathcal{V}} \in C_{\mathcal{V}(v)}$ with type $\mathcal{V}$, which leveraging promising properties of hyperbolic geometry.

The input of HSN is the node feature, whose norm could be out of the Poincaré ball defined in hyperbolic space. To make the node feature available in hyperbolic space, we use the exponential map to project the feature into the hyperbolic space, shown in (4). Specifically, let an Euclidean space vector $\mathbf{x}^E_i \in \mathbb{R}^d$ be the feature of node i, and then we regard it as the point in the tangent space $\mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ with the reference point $\mathbf{x}' \in \mathbb{H}^{d,c}$, using the exponential map to project it into hyperbolic space, obtaining $\mathbf{x}^{\mathcal{H}} \in \mathbb{H}^{d,c}$, which is defined as:

$$\mathbf{x}^{\mathcal{H}}_i = \exp^c_{\mathbf{x}'}(\mathbf{x}^R_i). \quad (7)$$

Then, We use meta-path guided random walks to obtain heterogeneous neighborhoods of a node [2]. Given an arbitrary meta-path $\mathcal{P} = \mathcal{V}_{v_1} \xrightarrow{\mathcal{E}_{e_1}} \mathcal{V}_{v_2} \xrightarrow{\mathcal{E}_{e_2}} \dots \xrightarrow{\mathcal{E}_{e_{n-1}}} \mathcal{V}_{v_n}$, our goal is to learn the semantically meaningful embeddings for all nodes under the constraint of meta-path $\mathcal{P}$. The transition probability at step i is defined as follows:

$$p(v^{i+1}|v^i_{\mathcal{V}_{v_i}}, \mathcal{P}) = \begin{cases} \frac{1}{|N_{\mathcal{V}_{v_{i+1}}}(v^i_{\mathcal{V}_{v_i}})|}, & (v^{i+1}, v^i_{\mathcal{V}_{v_i}}) \in E \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where $v^i_{\mathcal{V}_{v_i}}$ is node $v \in V$ with type $\mathcal{V}_{v_i}$, and $N_{\mathcal{V}_{v_{i+1}}}(v^i_{\mathcal{V}_{v_i}})$ denotes the $\mathcal{V}_{v_{i+1}}$ type of neighborhood of node $v^i_{\mathcal{V}_{v_i}}$.The meta-path guided random walk strategy ensures that the semantic relationships between different types of nodes can be properly incorporated into HSN.

The premise of network embedding models is to preserve the proximity between a node and its neighborhood. Therefore, in hyperbolic space, we use distances in Poincaré model to measure their proximity, as given in (6). We use a probability to measure the node $c_{\mathcal{V}}$ is a neighborhood of node v as following:

$$p(v|c_{\mathcal{V}}; \Theta) = \sigma[-d(u, v)] \quad (9)$$

where $\sigma(\cdot) = \frac{1}{1+exp(-x)}$ is an activate function. According to the (9), the object of HSN module is to maximize the probability as followings:

$$argmax \sum_{v \in V} \sum_{c_{\mathcal{V}} \in C_{\mathcal{V}}(v)} \log p(v|c_{\mathcal{V}}; \Theta) \quad (10)$$

C. Hyperbolic Temporal Network(HTN)

Historical information plays an indispensable role in temporal graph modeling since it facilitates the model to learn the evolving patterns and regularities. Although the latest hidden state H_{t-1} obtained by the recurrent neural network already carries historical information before time t, some discriminate contents may still be under-emphasized due to the monotonic mechanism of RNNs that temporal dependencies are decreased along the time span [13]. Inspired by [14], we design the hyperbolic temporal attention(HTA) unit generalizes H_{t-1} to the latest time window w snapshots $H_{t-w}, ..., H_{t-1}$, attending on multiple historical latent states to get a more informative hidden state. These historical states in the state memory are concatenate together and feed as input to the HTA unit, which is performed in tangent space due to its computational efficiency. Owing to the superiorities of attention, this unit fuses the final hidden state by figuring out the importance each

graph snapshots. The dataflow in the HTA unit is characterized by the following equations:

$$H_t^E = \log_{\mathbf{x}'}^c(H_t^{\mathcal{H}}) \tag{11}$$

$$H = concat(H_{t-w}^E, ..., H_{t-1}^E) \tag{12}$$

$$H_{t-1}^E = softmax(k^T \tanh(QH))H \tag{13}$$

$$H_{t-1}^{\mathcal{H}} = \exp_{\mathbf{x}}^c(H_{t-1}^E) \tag{14}$$

The learnable weight matrix Q and K are utilized to extract contextual information, where Q weights the node importance in each historical state and K determines the weights across the time windows.

Then, we use GRU, a variant of LSTM, as primary part of HTN to incorporate the current and historical node states, in view of the GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU gets rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.

HTN unit receives the sequential node embedding $X_t^{\mathcal{H}}$ from HSN and the hidden state $H_{t-1}^{\mathcal{H}}$ which is obtained from HTA. As sketched in Fig. 2, the input representations of HTN unit are assumed to sufficiently capture local structural information as well as attentive hidden state. The dataflow in the HTN unit is characterized by the following equations:

$$X_t^E = \log_{\mathbf{x}'}^c(X_t^{\mathcal{H}}), \tag{15}$$

$$H_{t-1}^E = \log_{\mathbf{x}'}^c(H_{t-1}^{\mathcal{H}}), \tag{16}$$

$$H_t^E = GRU(X_t^E, H_{t-1}^E), \tag{17}$$

$$H_t^{\mathcal{H}} = \exp_{\mathbf{x}}^c(H_t^E). \tag{18}$$

The main part of the unit is GRU. As the GRU is built in tangent space, we use the logarithmic transformations to project the $X_t^{\mathcal{H}}$ and $H_{t-1}^{\mathcal{H}}$ into tangent space. After processing representation using GRU, we project the embedding back to hyperbolic space. As we can see, the final embedding $H_t^{\mathcal{H}}$ fuses structural heterogeneity, content, and temporal information.

V. Optimization

Uniting the above modules, we formulate the learning objective from two aspects: topological learning and temporal evolution, corresponding to the following hyperbolic structural loss and hyperbolic temporal loss.

A. Hyperbolic Structural Loss

We leverage the negative sampling proposed in [15], which basically samples a small number of negative objects to enhance the influence of positive objects. The hyperbolic structure loss $\mathcal{L}(\Theta)$ aims to minimize the proximity between v and its neighborhood $c_{\mathcal{V}}$ while maximize the proximity between v and its negative sampled node n. The objective equation (9) can be formulated as following:

$$\mathcal{L}(\Theta) = \log \sigma[-d(\mathbf{x}_v, \mathbf{x}_{c_{\mathcal{V}}})] + \sum_{m=1}^{M} \mathbb{E}_{n^m \sim P(n)} \{\log \sigma[d(\mathbf{x}_v, \mathbf{x}_{n^m})]\} \tag{19}$$

where $P(n)$ is the pre-defined distribution from which a negative node n^m is drew from for M times. Our method builds the node frequency distribution by draw nodes regardless of their types.

B. Hyperbolic Temporal Loss

We build a hyperbolic temporal consistency consistency constraint $\mathcal{L}(t)$ on two consecutive time steps $(G_t, G_{t-1}))$, which is defined as:

$$\mathcal{L}(t) = \sum_{t=1}^{T} \sigma[d(\mathbf{x}_v, \mathbf{x}_{n^m})] \tag{20}$$

where the t denotes the loss is with respect to time step t.

C. The Unified Loss

To enable the learned representations to capture structural evolution, our objective function set the final loss function as:

$$\mathcal{L} = \mathcal{L}(\Theta) + \lambda \mathcal{L}(t) \tag{21}$$

where $\lambda \in [0, 1]$ is the hyper-parameter to balance the temporal smoothness and structural regularity. The final $\mathcal{L}$ not only minimizing the hyperbolic distance of a node with its connected nodes and maximizing with the sampled negative neighbors, but also minimizing the distance between the same node over two consecutive timestamps. As the parameters of DyHHE live in a Poincaré model which has a Riemannian manifold structure, it should be noted, the back-propagated gradient is a Riemannian gradient. It makes no sense Euclidean gradient based optimization works in this manifold. Therefore, we optimize $\mathcal{L}$ via Riemannian stochastic gradient descent(RSGD) optimization method [16]. The gradient of their distance can be derived as:

$$\Delta_v(d(\mathbf{x}_v, \mathbf{x}_{n^m})) = \frac{4}{\beta\sqrt{\gamma^2 - 1}}\left(\frac{\|\mathbf{x}_v\|^2 - 2\langle \mathbf{x}_v, \mathbf{x}_{n^m}\rangle}{\alpha^2}\mathbf{x}_v - \frac{\mathbf{x}_v}{\alpha}\right) \tag{22}$$

where $\alpha = 1 - \|\mathbf{x}_v\|^2$, $\beta = 1 - \|\mathbf{x}_{n^m}\|^2$, $\gamma = 1 + \frac{2}{\alpha\beta}\|\mathbf{x}_v - \mathbf{x}_{n^m}\|^2$.

VI. Experiments and Analysis

In this section, we conduct extensive experiments with the aim of answering the following research questions:

- **RQ1** How does DyHHE perform.
- **RQ1** What does each component of DyHHE bring?

A. Datasets

To evaluate the effectiveness of our model, we conduct experiments on two datasets from real-world platforms. The datasets are summarized in Table I

- **DBLP** is a database of publications. Specifically, we collected the papers from four research areas which contains three types of nodes, i.e., author(A), paper(P), venue(V) and two types of edges, i.e., author-paper(write), paper-venue(publish). Timestamps denote the year of the publication.
- **MovieLens** [17] contains knowledge about movies. MovieLens users from the late 1990s to the early 2000s.

We extract a subset of MovieLens, which contains three types of nodes, i.e., actor(A), movie(M), and director(D) and two types of edges, i.e., actor-movie(act in) and director-movie(direct).

TABLE I
STATISTICS OF DATASETS

DBLP	A	P	V	AP	PV	Snapshots
	14475	14376	20	41794	14376	16
MovieLens	A	M	D	AM	MD	Snapshots
	11718	9160	3510	64051	9160	13

Data with power-law structure can be naturally modeled in hyperbolic spaces [6]. Therefore, we use two real-world HINs datasets which have been proved to conform to the power-law distribution of nodes [18], [19].

B. Baselines

We present comparisons against several static graph embedding methods to analyze the gains of using temporal information for link prediction. To ensure a fair comparision, we also conduct experiments on several heterogeneous network representation model to further demonstrate the superiority of the proposed model DyHHE. Moreover, we also compare to the hyperbolic embedding model, HGCN and PoincareEmb.

- **Node2vec** [20] is a static embedding method to generate vector representations of nodes on a graph. It learns low-dimensional representations for nodes in a graph through the use of random walks.
- **Metapath2vec** [2] is an HIN embedding method. It learns feature representations by capturing node pairs within w-hop heterogeneous neighborhood via meta-path guided random walks in the network.
- **HGCN** [11] is a static embedding method which leverages both the expressivenees of GCNs and hyperbolic geometry to learn node reoresentations for hierachical and scale-free graphs.
- **DySAT** [4] computes node representations by jointly employing self-attention layers along two dimensions: structural neighborhood and temporal dynamics.
- **HDGAN** [5] is based on three levels of attention, namely structural-level attention, semantic-level attention and time-level attention and attempts to use the attention mechanism to take the heterogeneity and dynamics of the network into account at the same time, so as to better learn network embedding.
- **PoincareEmb** [21] is a method that preserves proximities of node pairs linked by an edge via embedding network into a Poincaré ball.

For random walk based methods like Node2vec and Metapath2vec, we set neighborhood size to 5, walk length to 80, ignoring the temporal regularity. As for meta-path guided random walks like metapath2vec and PoincareEmb, we use meta-path "A–P–V–P–A" in DBLP and "A-M-D-M-A" in MovieLens. For dynamic homogeneous baselines Dysat, we treat events as homogeneous. The train/test ratio is set to 80%/20%.

C. Link Prediction Comparison(RQ1)

Link prediction is to predict the type $\mathcal{V}$ interaction at time step t, which can be used to test the generalization performance of a network embedding method. Given all temporal heterogeneous events before time step t and two nodes u and v. For each type of edge, we treat all events at time t as the positive link, and an equal number of negative examples in the training set are created by sampling the node pairs not interconnected. Subsequently, we split the chosen edges and negative samples into validation and test. In our experiments, we test the models regarding their ability of correctly classifying true and false edges by computing average precision (AP) and area under the ROC curve (AUC) scores. We uniformly train both the baselines and DyHHE by using early stopping based on the performance of the training set.

TABLE II
AUC SCORES OF LINK PREDICTION RESULT.

	DBLP		MovieLens	
Edge	**A-P**	**P-V**	**A-M**	**M-D**
Node2vec	85.32 ± 0.7	85.25 ± 0.8	81.27 ± 0.2	83.44 ± 1.1
Metapath2vec	87.46 ± 0.9	88.11 ± 0.9	82.3 ± 0.1	81.57 ± 1.4
HGCN	89.8 ± 1.2	90.27 ± 0.4	85.4 ± 0.2	85.17 ± 1.3
DySAT	90.66 ± 0.2	90.21 ± 0.4	87.32 ± 0.3	86.75 ± 0.9
HDGAN	87.42 ± 0.4	88.66 ± 0.6	85.78 ± 0.9	86.12 ± 0.7
PoincareEmb	87.85 ± 0.4	87.16 ± 0.3	86.71 ± 1.7	85.63 ± 0.9
DyHHE	$\mathbf{92.69 \pm 0.4}$	$\mathbf{93.19 \pm 0.3}$	$\mathbf{90.13 \pm 0.4}$	$\mathbf{90.77 \pm 0.3}$

TABLE III
AP SCORES OF LINK PREDICTION RESULT.

	DBLP		MovieLens	
Edge	**A-P**	**P-V**	**A-M**	**M-D**
Node2vec	86.94 ± 0.4	86.78 ± 0.6	83.17 ± 0.3	82.15 ± 0.9
Metapath2vec	87.83 ± 1.2	86.43 ± 0.7	81.77 ± 0.4	82.72 ± 0.3
HGCN	88.6 ± 0.7	89.33 ± 0.5	84.6 ± 1.2	84.45 ± 1.1
DySAT	90.37 ± 0.4	90.71 ± 0.3	85.72 ± 0.6	85.25 ± 0.3
HDGAN	89.61 ± 0.3	87.74 ± 0.6	86.33 ± 1.1	85.12 ± 0.9
PoincareEmb	88.15 ± 1.1	86.29 ± 0.2	84.12 ± 0.8	85.81 ± 0.7
DyHHE	$\mathbf{92.13 \pm 0.7}$	$\mathbf{93.49 \pm 0.5}$	$\mathbf{91.02 \pm 0.6}$	$\mathbf{89.43 \pm 0.8}$

We repeat each experiment five times and report the average value with the standard deviation on the test sets in Table II and Table III. It is observed our model achieves the best results and has a more than $4-6\%$ AUC and AP improvement comparing to the best baseline across all datasets. First of all, the Metapath2vec has a better performance than Node2vec which means the advantage of the proper consideration and accommodation of the network heterogeneity. Despite the existence of multiple types of nodes and edges in heterogeneous graph, Metapath2vec performs pooly compared with the dynamic methods DySAT and HDGAN, which confirms the importance of temporal regularity in dynamic graph modeling. Moreover, the performance gap between DyHHE and HDGAN suggests that the significantly benefit from hyperbolic geometry. It is

TABLE IV
ABLATION STUDY(AUC).

	DBLP		MovieLens	
edge	**A-P**	**P-V**	**A-M**	**M-D**
No Hyperbolic	89.03 ± 0.5	90.72 ± 0.4	85.34 ± 0.3	84.79 ± 0.5
No Temporal	89.18 ± 0.5	89.14 ± 0.6	89.26 ± 0.4	90.81 ± 0.6
Original	$\mathbf{92.69 \pm 0.4}$	$\mathbf{93.19 \pm 0.3}$	$\mathbf{92.13 \pm 0.4}$	$\mathbf{90.77 \pm 0.3}$

worth mentioning that HGCN and PoincareEmb also has not bad performance despite being agnostic to semantic relationships and temporal information in heterogeneous graph, which indicates further improvements to DyHHE on transforming embeddings from Euclidean space to Hyperbolic space.

D. Ablation Study (RQ2)

To investigate the superiority of the main components of our model, we conduct an ablation study by independently removing the hyperbolic geometry and temporal modules from DyHHE to create simpler architectures. And we compare DyHHE with different variants on DBLP and MovieLens datasets. When we remove the hyperbolic geometry and build the model in Euclidean space, the HSN and HTN units are converted to the corresponding Euclidean space. We show the variant models results in Table IV. From the results, we observe that in MovieLens the removal of hyperbolic geometry consistently deteriorates performance, while the DBLP only declines about 4%. One major explanation is that the MovieLens has a high-hyperbolicity, which indicates the dataset has a more evident hierarchical structure. And the hierarchical structure and tree-like data can naturally be represented and preserved by hyperbolic geometry. The effect of temporal module is also significant because of the performance degradation by removing the temporal block. This observation conforms to the nature of graph evolution since the behaviors usually have periodical patterns such as recurrent links or communities. In summary, DyHHE generates more appropriate embeddings for dynamic heterogeneous neetwork than comparative baselines, suggesting its ability to capture and incorporate the underlying structural and temporal information.

VII. CONCLUSIONS

In this work, we introduce a novel hyperbolic geometry-based node representation learning framework in dynamic heterogeneous networks in which there exists diverse types of nodes and links. To address the network heterogeneity and temporal evolution, we propose the DyHHE model. In general, DyHHE computes dynamic node representations through maximize proximity in consideration of multiple types of neighborhoods for a given node and follow the effective GRU framework by leveraging the superiority of hyperbolic graph neural network. Our experimental results on two real-world datasets indicate significant performance gains for DyHHE over several static and dynamic heterogeneous graph embedding baselines. An interesting future direction is generalizing our method to more challenging tasks.

REFERENCES

[1] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.

[2] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[3] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[4] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 519–527.

[5] Q. Li, Y. Shang, X. Qiao, and W. Dai, "Heterogeneous dynamic graph attention network," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2020, pp. 404–411.

[6] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.

[7] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[8] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2857–2866.

[9] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[10] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," *arXiv preprint arXiv:1910.12892*, 2019.

[11] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," *Advances in neural information processing systems*, vol. 32, pp. 4868–4879, 2019.

[12] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention lstm networks for 3d action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1647–1656.

[14] Q. Cui, S. Wu, Y. Huang, and L. Wang, "A hierarchical contextual attention-based network for sequential recommendation," *Neurocomputing*, vol. 358, pp. 141–149, 2019.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[16] S. Bonnabel, "Stochastic gradient descent on riemannian manifolds," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2217–2229, 2013.

[17] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[18] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5337–5344.

[19] Y. Zhang, X. Wang, N. Liu, and C. Shi, "Embedding heterogeneous information network in hyperbolic spaces," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 2, pp. 1–23, 2021.

[20] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[21] Z. Huang and N. Mamoulis, "Heterogeneous information network embedding for meta path based proximity," *arXiv preprint arXiv:1701.05291*, 2017.

Using Multi-feature Embedding towards Accurate Knowledge Tracing

Yang Yu, Caidie Huang, Liangyu Chen, Mingsong Chen
MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application
East China Normal University, Shanghai, China
lychen@sei.ecnu.edu.cn, mschen@sei.ecnu.edu.cn

***Abstract*—Knowledge tracing is a crucial task in intelligent tutoring systems. Aiming at the shortcomings of traditional knowledge tracing technology such as low prediction accuracy, overfitting and low utilization of multi-features, this paper proposes a knowledge tracing model SRGCA-M using multi-feature embedding with stacked residual GRU network. Compared with the traditional methods that only use the historical record of answering exercises, our approach utilizes a variety of features in the learning process of students to deep characterize students' learning. We increase the layers number of GRU network to expand the capacity of sequence learning and use residual connections to solve the problems of network degradation and vanishing gradient. We use the auto-encoder to solve the problem that the cross-feature encoding will rapidly increase the dimension of the input data. Comprehensive experimental results demonstrate that compared with various advanced techniques, our approach can not only achieve better performance of tracking knowledge changes of students but also fully utilize multi-feature information of students in the learning process.**

***Index Terms*—Knowledge Tracing; GRU; Multi-feature Embedding; Auto-Encoder**

I. Introduction

Intelligent tutoring systems (ITS) are computer-based educational systems that act as smart teachers to guide students' learning. Knowledge tracing is the key step in ITS to track the change process of the knowledge mastery of students according to the historical records of learning, predict their future learning performance, and better provide students with personalized learning guidance services. Some pedagogical researchers believe that the knowledge concepts investigated in the exercise may be specific and relevant, and the mastery of the knowledge investigated in the exercise will affect their performance in the exercise, which means that the exercise investigation is the manifestation of the cognitive state of students.

With the popularization of online education, a large number of exercise answering data of students have been generated on the Internet, including knowledge concept of exercise, students' answer scores, students' answer practice, students' answer times, etc. Enough data promote the research progress of knowledge tracing models. However, there are still some problems, such as inaccurate prediction results, slow convergence speed and low utilization of multi-features. These problems limit the application and promotion of knowledge tracing in education to a certain extent. Researches have shown that a variety of features are helpful for evaluating students and personalizing instruction. Therefore, the research on knowledge tracing can not only promote the development of knowledge tracing in the education industry, but also reduce the stress of teachers and improve learning efficiency of students. Therefore, the research on knowledge tracing is of great significance to intelligent education.

DOI reference number:10.18293/SEKE2022-142

In order to improve the prediction accuracy of the knowledge tracing model, this paper proposes a Stacked Residual Gate Recurrent Network using Multiple Features with Cross Encoding and Auto-encoder (SRGCA-M), which can use various data in the process of students' answering. First, we use the lightGBM algorithm to select the features with high importance, and the cross-feature encoding and one-hot encoding method to encode the selected features. Because cross-feature encoding will rapidly increase the dimension of input features, we utilize an auto-encoder to compress the input features, and the compressed features are input into the SRG for training and prediction. SRGCA-M is tested on the Riiid dataset and compared with lightGBM and DKT. In addition, three ablation experiments are executed to verify the effects of multi-feature encoding and auto-encoder compression. The results show that the SRGCA-M achieves the best performance. This paper makes the following three major contributions:

- To improve model performance, we make full use of multiple features in the learning process of students by utilizing lightGBM feature selection, multi-feature cross-encoding and auto-encoder. In contrast, traditional knowledge tracing methods utilize the historical answer records of exercises.
- We increase the layers of network to expand the capacity of sequence learning by using the stacking GRU network. Besides, the use of residual connections solve the problems of network degradation and gradient vanishing.
- We improve the performance of the model by using an auto-encoder to represent the features which addresses the problem that cross-feature encoding will rapidly increase the dimensionality of the input data.

The rest of this paper is organized as follows. Section 2 summarizes the related works on knowledge tracing. Section 3 demonstrates our SRGCA-M model. Section 4 presents the experimental results and discussion. Finally, Section 5

concludes the paper.

II. Related Works

In order to solve the problems of low accuracy of knowledge tracking and low utilization of multi-features, various knowledge tracking methods have been proposed. For example, Bayesian Knowledge Tracing (BKT) [1] is a typical model based on probability graph. The model uses a set of binary variables to model the knowledge space characteristics of students, and each variable represents whether students master some knowledge concepts. The knowledge tracing model based on probability graph uses pedagogical theory, which is highly interpretable. However, the prediction efficiency largely depends on the rationality of establishing the probability map. When the establishment of the probability map is not reasonable, the performance will be greatly reduced.

Knowledge Proficiency Tracing (KPT) model [2], based on Probabilistic Matrix Factorization (PMF) [3] is proposed for knowledge tracing task. This model can effectively improve the prediction performance using the expert-marked topic knowledge concept matrix (Q matrix). However, the matrix decomposition model cannot add relevant information other than the topic knowledge concepts, such as exercise discrimination and exercise difficulty.

The Deep Knowledge Tracing (DKT) model proposed by Piech et al. [4] is the basic model in the field of deep knowledge tracing, which is based on the Recurrent Neural Network (RNN). The prediction performance of DKT is better than the classical methods at that time [5]. However, the DKT model suffers from poor interpretability, long-term dependencies, and few learning features [6]. In order to solve these problems, many researchers are committed to in-depth research on DKT and put forward many new methods. Dong et al. [7] used Jaccard coefficient to calculate the attention weight between knowledge components in the model a-dkt, and combined LSTM and total attention value to get the final prediction result. Zhang et al. [8] used the method of feature engineering to add the dimension reduction of answer time, answer times and the first action to the input layer of LSTM by using an auto-encoder.

For many years, the knowledge tracing method based on RNNs [9] has been dominant for the following reasons: i) the method based on neural network does not need to select features manually; ii) online education platforms generate massive amounts of data. The score data of exercises is the most relevant explicit data in students' knowledge space, and it is also easy to obtain. Therefore, the neural network method based on score data is universal; iii) the calculation of RNN and its variants combines the output of past time information and integrates them into the calculation of current time. Therefore, in knowledge tracing, this kind of model can achieve good results on datasets with temporal information.

Therefore, in this paper, we propose a residual network based on GRU, which uses a stacked residual GRU network (SRG) to learn students' answer sequences, and uses residual connections to reduce the difficulty of model training.

III. Our Method

A. Problem Definition

The task of knowledge tracking is to track students' knowledge mastery level and predict their future performance according to the historical records of answering exercises. The input is represented by the following formula,

$$\begin{aligned} &D = \{S_1, S_2, \ldots, S_N\}, \\ &S_i = \{x_{i1}, x_{i2}, \ldots, x_{it}, \ldots, x_{iM}\}, \\ &x_{it} = \{q_{it}, a_{it}\}. \end{aligned} \tag{1}$$

Suppose that there are N students, each student answers M exercises and $S_i = \{x_{i1}, x_{i2}, \ldots, x_{it}, \ldots, x_{iM}\}$ makes a exercise sequence for the students, and D represents the student set. At time t, x_{it} contains two parts: i) the exercise that the student is answering at the current moment; ii) the learner's answer to the exercise a_{it}. When x_{it} is 0, it means that the student answered incorrectly, and 1 means that the student answered correctly. The student behavior sequence is encoded and input into the recurrent neural network for training, and the learner's knowledge mastery level is obtained through the prediction output layer. Finally, the correct rate of the student's answer at the next moment is predicted. The range is 0 to 1, indicating the prediction probability. There are two basic tasks of knowledge tracking: i) predict students' future answer performance, i.e., the correct answer rate in the next time step; and ii) track the changes of students' mastery of knowledge concept to facilitate personalized learning guide.

B. Overall Structure

The framework of SRGCA-M model is shown in Figure 1 and consists of the following parts: data preprocessing, feature selection, multi-feature encoding and deep learning prediction. In the data preprocessing stage, the original dataset needs to be cleaned and sorted to get a relatively complete dataset. In the feature selection stage, we use lightGBM algorithm to calculate the importance of features, obtain the feature importance ranking, and predict the students' scores as a comparative experiment. In the multi-feature encoding stage, cross-feature encoding and one-hot encoding are used to encode the selected important features and students' response features. The encoded features are compressed by auto-encoder (AE). Finally, in the deep learning prediction stage, the compressed feature is integrated into SRG model to track students' knowledge level and predict students' achievement.

C. Multi-Feature Selection

Saivastava et al. [10] pointed out that the performance of models using cross-features is improved compared to models using single feature. Inspired by this, we utilize lightGBM to process multi-feature data. Firstly, input the students' muti-feature answer data, use the histogram algorithm to find the feature with the maximum gain, and determine the optimal segmentation point of the decision tree according to the feature. Using leaf-wise leaf growth strategy with depth constraints to generate cart tree. Then calculate the residual

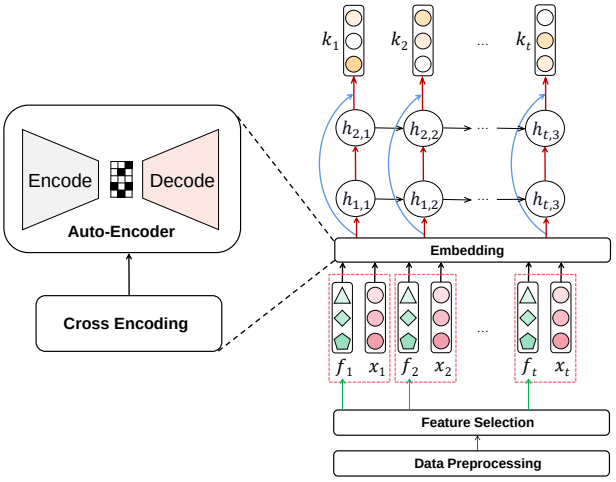


Fig. 1. SRGCA-M model framework.

value of cart tree, take the residual result of the previous tree as the training sample, train the next cart tree and repeat the training. Finally, the cart tree generated by each training round is weighted and summed to obtain the final prediction model.

LightGBM [11] algorithm measures the importance of feature attributes based on the number of times the feature is used as segmentation points. Sort the feature elements from large to small according to the attribute importance. Search from the complete set of all features, and judge whether to delete the feature with the lowest importance according to the result accuracy. Traverse all features and output the optimal feature subset. The input of the algorithm are the dataset D, the feature set $F = \{T_i \mid i = 1, 2, \ldots, d\}$, and the output is the optimal feature subset F_{best}.

D. Multi-Feature Encoding

After the features of the dataset are filtered by lightGBM algorithm, multiple features need to be encoded to form the input data. In this paper, we pose three multi-feature embedding methods: direct concatenating method, cross-feature encoding method and compressed cross-feature encoding method. Next, three embedding methods are introduced in detail.

Direct concatenating method forms a new vector by concatenating the answer data and the optimal feature directly. This method can simply convert a single feature vector into multiple feature vectors, which is the input x_t of the model.

Crossed features refer to the cross-encoding result of student answers and selected multi-features. The The cross-feature encoding method can be expressed by the following formula,

$$\begin{aligned} C\left(s_t, c_t\right) &= s_t + [\max(s) + 1] * c_t, \\ x_t &= O\left(C\left(s_t, c_t\right)\right) \oplus O\left(C\left(F_t, c_t\right)\right) \oplus O\left(F_t\right), \end{aligned} \tag{2}$$

where C represents cross encoding, O represents one-hot encoding, $\oplus$ sign indicates the concatenation of two vectors, i.e., $C = A \oplus B$, which indicates that vector B is spliced at the end of vector A. The number of rows of vector C is the same as that of A and B. The number of columns of vector C is the sum of that of A and B.), s_t represents the ID of the knowledge concept, and c_t represents the result of answer (1 means correct, 0 means wrong), F_t represents the optimal features selected by the LightGBM algorithm.

Because the cross-feature encoding will lead to the rapid increase of input dimension, the compressed cross-feature encoding method used in SRGCA-M utilizes an auto-encoder to compress the cross-encoded features. The specific calculation method is listed as follows,

$$\begin{aligned} z_t &= E\left(x_t\right) = \sigma\left(W x_t + b\right), \\ x_t' &= D\left(z_t\right) = \sigma\left(W x_t + b\right), \end{aligned} \tag{3}$$

where x_t is the input, the function E represents the encoding operation, and the function D represents the decoding operation. z_t represents the learned latent variable, which can be used as input data.

E. Stacked Residual GRU Network (SRG)

This paper uses the stacked residual GRU network to deal with the deep knowledge tracing task [12]. It improves the performance of traditional recurrent neural networks by increasing the number of network layers to expand the capacity of sequence learning. Besides, the residual connections help solve the problems of network degradation and gradient vanishing. The stacked residual GRU network SRG can be defined by the following formula,

$$\begin{aligned} h_{1,t} &= f_{gru-1}\left(h_{1,t-1}, x_t\right), \\ h_{2,t} &= f_{gru-2}\left(h_{2,t-1}, h_{1,t}\right), \\ k_t &= \sigma\left(W_{kt} h_{2,t} + b_k\right). \end{aligned} \tag{4}$$

The input x_t enters the first layer of GRU network to obtain the hidden variables $h_{1,t}$. Then the output of the first layer is used as the input of the second layer to obtain the output $h_{2,t}$ of the second layer GRU network. Then the knowledge level vector k_t is obtained from the full connection layer.

Because the increase of layers of the recurrent neural network will make it challenging to fit the model training, the SRG model introduces residual connection [13]. The addition of residual connections can make the training of stacked GRU network converge easily, so this paper proposes a stacked residual GRU network *SRG* with residual connections.

$$k_t = \sigma\left(W_{kh}\left(h_{2,t} \oplus x_t\right) + b_k\right). \tag{5}$$

Equation (5) reflects this idea by concatenating the input x_t of the model with the output $h_{2,t}$ of the hidden layer. Then input the concatenated vector to the next layer for prediction. The loss function is defined as follows:

$$\begin{aligned} L = &-\sum_t \left(a_t \log k_{t+1}\left(q_t\right) + \left(1 - a_t\right) \log\left(1 - k_{t+1}\left(q_t\right)\right)\right) \\ &-\sum_t \left(x_t \log x_t' + \left(1 - x_t\right) \log\left(1 - x_t'\right)\right). \end{aligned} \tag{6}$$

The whole loss is divided into two parts. The first part is SRG loss and the second part is the loss function of the auto-encoder, which also uses the cross-entropy loss function. x' is the reconstructed data generated from the encoder.

IV. EXPERIMENTS

To evaluate the effectiveness of our approach, we implement SRGCA-M based on Pytorch framework in this section. We evaluate and compare the effectiveness of SRGCA-M with other methods. All the experiments are conducted on a workstation computer with Centos 7 operating system, Intel i7-9700K CPU with 16 GB memory, and NVIDIA GRX2080Ti GPU with 11 GB memory.

Formally, we design substantial experiments to answer the following two research questions.

RQ_1**: (Effectiveness of multi-feature):** What is the performance of SRGCA-M using multiple features cross encoding and auto-encoder compared with SRG-S using single feature?

RQ_2**: (Effectiveness of cross encoding and auto-encoder):** What is the performance of SRGCA-M using multiple features cross encoding and auto-encoder compared with SRG-M using multiple features and SRGC-M using multiple features and cross encoding?

A. Dataset Configurations

The dataset used in the experiments is *Riiid*, which is derived from Riiid Answer Correctness Prediction, a student performance prediction competition on the Kaggle website. The Riiid dataset provides historical learning records of students, other students' performance on the exercises and other metadata of the exercises. All Riiid data are divided into three files: train.csv, questions.csv and collections.csv. The details of Riiid dataset are shown in Table I. In the experiments, the dataset is divided into training set and verification set according to the ratio of $4:1$.

TABLE I
STATISTICAL INFORMATION OF DATASET.

Dataset	Students	Knowledge Concepts	Records	Answers/Person
Riiid	174,954	187	41,667,551	238

Table II describes the specific field content of the questions.csv file. This file contains the relevant information of the question, such as id, correct answer and corresponding knowledge concepts. In the process of data preprocessing, it is necessary to compare the students' answer records in train.csv with the correct answers to the exercises in questions.csv to determine whether the students answered correctly.

TABLE II
QUESTIONS DATA CONTENT DESCRIPTION.

Field name	Field Description
question_id	ID of the problem
bundle_id	ID of the problem set
correct_answer	Right key
part	Relevant parts of TOEIC test
tags	One or more detailed label codes

The *train.csv* file contains multi-feature information about the exercises and the learning process of students. It includes whether the exercises are answered correctly, the time it takes to answer the exercise, the historical answering time, the time from the first interaction of students to the completion of the exercise, and the average time it takes to answer the previous set of exercises, whether students viewed explanations and correct answers after answering the previous set of exercises. Note that the students' learning is divided into two forms: watching lectures and answering exercises, and the information of watching lectures should be ignored in the records.

B. SRGCA-M Using Multiple Features

1) Experimental Settings: The baseline model in this experiment is lightGBM. At the same time, three groups of ablation experiments are set according to the characteristics, namely single feature SRG model (SRG-S), multi-feature SRG model (SRG-M) and multi-feature cross-encoding SRG model (SRGC-M).

LightGBM: lightGBM is the feature selection algorithm used in this experiment. LightGBM predicts students' scores through these feature training models. The results can get the importance ranking of features to the results. According to the feature importance ranking of lightGBM, the top-ranked features are selected for multi-feature encoding.

SRG-S: SRG-S is a single-feature SRG model. It uses a stacked residual GRU network to predict students' grades. The model is a single-feature model, which only predicts future grades through knowledge concepts and historical answer records of students.

SRG-M: SRG-M is a multi-feature SRG model based on SRG-S, where additional features are highly important features extracted from the lightGBM experimental results. SRG-M encodes students' answer records and additional features as the input of the prediction model. The model simply concatenates the features without cross-feature encoding or multi-feature compression.

SRGC-M: SRGC-M is a cross multi-feature SRG, which performs one-hot encoding and cross-feature encoding on student answer records and additional features, and takes the fused features as the input of the prediction model. The model adds cross-feature encoding based on SRG-M. There is no multi-feature compression like SRG-M.

SRGCA-M: SRGCA-M is a compressed cross-encoded multi-feature SRG proposed in this paper. The model compresses student answer records and additional features after one-hot encoding and cross encoding as the input of the prediction model. Based on the SRGC-M, an auto-encoder is used to compress the cross-encoded multi-features. Then the compressed latent variables are input into the prediction network SRG for training.

2) Parameter Settings: Parameters of LightGBM are set as follows. The number of leaves is 200. When building a weak learner, the proportion of random sampling of features is 0.75, the sampling frequency is set to 10, bagging fraction is set to 0.8, the maximum number of iterations is set to 10000. The early stop round is set to 10 which means stop the training if the 10 training times are not optimized, verbose evaluation is set to 50, which means information is output every 50 iterations. The learning rate of SRGCA-M is set to 0.001, the

maximum step size is set to 50, which means every 50 records are a set of input data, less than 50 data are filled with 0, the minimum batch number is set to 128.

We set up three ablation experiments to verify the effectiveness of the model, where each ablation model verifies the effectiveness of a corresponding module. The learning rate is set to 0.001, the maximum step size is set to 50, which means every 50 records are a group of input data, less than 50 data are filled with 0, the minimum batch number is set to 128. SRGCA-M model and its ablation experimental model are optimized by Adam optimizer. All models use the evaluation functions AUC, RMSE and F1 as performance evaluation metrics.

3) Feature Selection: We calculate new features from the original data after cleaning raw data and selecting four million answer records with relatively complete data. After the training of lightGBM algorithm, we get the importance ranking of these features, shown in figure 2. These features are ranked as follows: the interval time of students answering the exercises for the first time, the average correct rate of exercises, and the average correct rate of students, the interval time of students answering the exercises for the third time, average answering time of the exercises, average number of times to check the problem analysis, average answering time of students, time for students to answer the previous set of exercises, the interval time for students to look back after answering wrong exercises, the interval time of students answering the exercises for the second time, the average number of times students viewed the problem resolution and correct answers, the top category code of lecture, the number of correct answers, and whether students viewed the resolution and correct answers after answering the exercises. We remove the last three features with obvious low scores and select the remaining features which are useful for evaluating the knowledge level of students and predicting their future performance.

4) Student Achievement Prediction: The result of student achievement prediction is crucial in the performance evaluation of knowledge tracing task. In this experiment, lightGBM method is chosen as the comparative baseline, and the models SRG-S, SRG-M and SRGC-M are designed in the ablation experiment. The Riiid dataset is used in the experiment. We use the data after feature selection. The dataset is divided into training set and test set according to $4:1$. The *AUC*, *RMSE* and *F1* scores of each model on the Riiid dataset are recorded respectively.

Table III shows experimental results for different methods and Fig.3 compares them in visualization. One can easily observe that SRGCA-M achieves the highest AUC value and F1 score, while the RMSE value is also the lowest. Obviously, the prediction performance of SRG-based methods is much better than that of LightGBM. Except that the performance of SRGC-M using cross-feature encoding is lower than that of LightGBM, the performance of other SRG-based models is better than that of LightGBM. The performance of SRG-S using a single feature is similar to that of SRG-M, and slightly lower than the performance of SRGCA-M. In addition, The effect of SRG-M is worse than that of single-feature SRG-S. After adding additional features to SRG-M, the input dimension is increased by 11 times, so the training of the model will be more difficult. What's more, the performance of SRGC-M using cross-feature encoding is worse than the simply connected SRG-M. After adding additional features and crossed features to SRGC-M, the input dimension is increased by 22 times, so there are too many network parameters. Instead, the model becomes bloated and more difficult to converge. Importantly, after using the auto-encoder to compress the cross-multiple features, SRGCA-M has a great improvement than SRGC-M in the prediction performance compared , and the AUC value is improved by more than 16%.

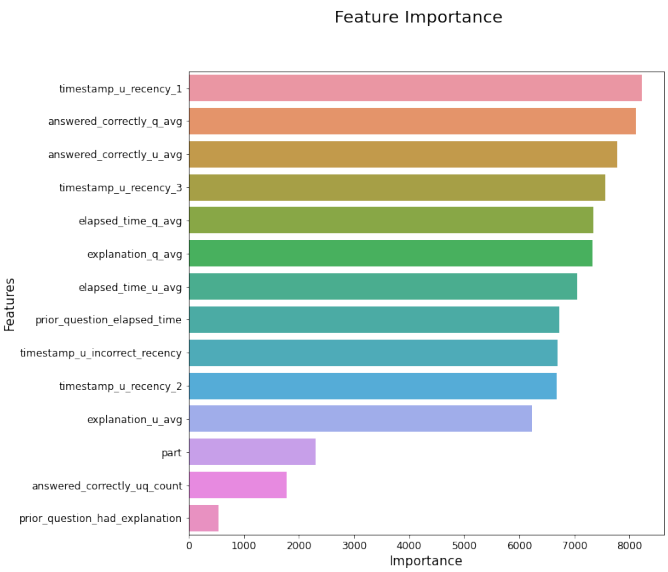


Fig. 2. Results of feature selection.

TABLE III
TEST RESULTS OF EACH MODEL ON RIIID DATASET.

Model	AUC	RMSE	F1
LightGBM	0.778	0.423	0.771
SRG-S	0.866	0.298	0.831
SRG-M	0.818	0.411	0.795
SRGC-M	0.708	0.529	0.704
SRGCA-M	**0.868**	**0.296**	**0.833**

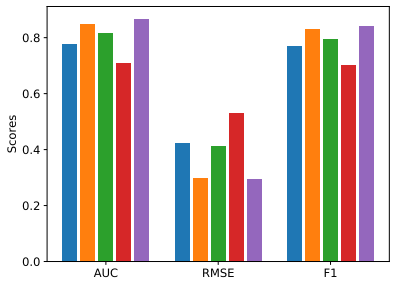


Fig. 3. Comparison of each model on Riiid Dataset.

In addition to predicting the correct rate of students answering questions at the next moment, another task of knowledge tracking is to track the change of students' knowledge level. Figure 4 is the visualization result of the data randomly selected from the Riiid validation set and predicted by the SRGCA-M model.

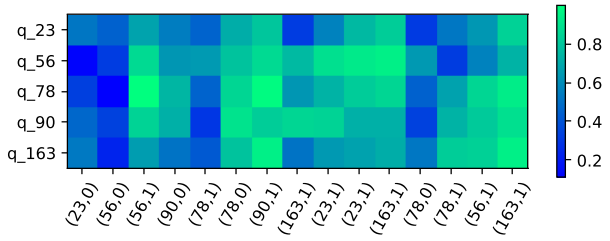


Fig. 4. Heatmap of SRG model tracking knowledge changes.

The horizontal axis represents the time series of a student answering questions. A two-tuple is used to represent the answer records. For example, the first record (23,0) represents the student's answer to question No. 23, and the answer is wrong. The vertical axis represents questions answered by students. The color of each square in the figure represents the student's mastery of knowledge at the current moment. The darker the color, the worse the student's mastery of the knowledge point corresponding to the question.

The color of each knowledge concept changes at different times, which indicates that the mastery level of knowledge concept is also changing accordingly. Focus on the first row, it is the changing process of students' mastery of knowledge concept No. 23. At the first moment, the student answered the question incorrectly, so the color of the corresponding square is dark. At the ninth and tenth moments, the student answered the question correctly, so the color of the square becomes light, and the color of the tenth moment is light. The change process of other knowledge concepts also has similar rules. From the results shown in Figure 4, we can sum up that the SRGCA-M model can effectively track the changes of mastery level of knowledge, which helps students to provide personalized tutoring services for learning guidance.

V. Conclusion

With the advent of the information age, the demand of people for online education is continuously increasing. As a key technology in intelligent tutoring systems, knowledge tracing has attracted many attentions. Although knowledge tracing technology has made great progress, there are still some problems. Aiming at the problems of inaccurate prediction results, slow convergence speed and low data utilization in knowledge tracing technology, this paper proposes a multi-feature knowledge tracing model SRGCA-M, which can effectively use the multi-feature information of students' learning history. SRGCA-M model first uses the lightGBM algorithm to filter student features for selecting the features with high importance to the results. The important features and historical answers of students are coded by cross-feature encoding method and one-hot encoding method. Because the feature dimension after encoding is too high, auto-encoder is used to compress the feature for better performance. Finally, the knowledge tracing model SRG is used to predict students' future performance and track students' knowledge mastery level. The experimental results show that the SRGCA-M model surpasses the lightGBM and DKT related models in prediction performance, and also gets better performance than other models in ablation experiments, which shows that our model can better track the knowledge level of students.

Acknowledgment

This work was supported by Natural Science Foundation of China 61872147 ,Shanghai Trusted Industry Internet Software Collaborative Innovation Center and Open Research Fund of Engineering Research Center of Software/Hardware Codesign Technology and Application, Ministry of Education (East China Normal University). Mingsong Chen and Liangyu Chen are the corresponding authors.

References

[1] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized bayesian knowledge tracing models," in *International Conference on Artificial Intelligence in Education (AIED)*. Springer, 2013, pp. 171–180.

[2] Y. Chen, Q. Liu, Z. Huang, L. Wu, E. Chen, R. Wu, Y. Su, and G. Hu, "Tracking knowledge proficiency of students with educational priors," in *Proceedings of Conference on Information and Knowledge Management (CIKM)*, 2017, pp. 989–998.

[3] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2008, pp. 1257–1264.

[4] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *ArXiv Preprint arXiv:1506.05908*, 2015.

[5] M. Khajah, R. V. Lindsey, and M. C. Mozer, "How deep is knowledge tracing?" *ArXiv preprint arXiv:1604.02416*, 2016.

[6] J. Lee and D.-Y. Yeung, "Knowledge query network for knowledge tracing: How knowledge interacts with skills," in *Proceedings of the International Conference on Learning Analytics & Knowledge (LAK)*, 2019, pp. 491–500.

[7] D. Liu, H. Dai, Y. Zhang, Q. Li, and C. Zhang, "Deep knowledge tracking based on attention mechanism for student performance prediction," in *Proceedings of International Conference on Computer Science and Educational Informatization (CSEI)*. IEEE, 2020, pp. 95–98.

[8] L. Zhang, X. Xiong, S. Zhao, A. Botelho, and N. T. Heffernan, "Incorporating rich features into deep knowledge tracing," in *Proceedings of ACM Conference on Learning @ Scale (L@S)*, 2017, pp. 169–172.

[9] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, pp. 3146–3154, 2017.

[12] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *Proceedings of International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

A Zero-Shot Relation Extraction Approach Based on Contrast Learning

Hongyu Zhu, Jun Zeng, Yu Yang, Yingbo Wu
School of Big Data & Software Engineering
Chongqing University
Chongqing, China
zengjun@cqu.edu.cn

Abstract—The most significant advantage of the unseen relation extraction is that it can recognize unlabeled relations. While Zero-Shot Learning can meet the requirements of the identification of unseen relation through relation description information without labeled datasets. However, unseen relation extraction requires an effective method in representation and generalization, which become a challenge for zero-shot learning approach. In this paper, we propose a Zero-Shot learning Relation Extraction based on Contrastive learning Model (ZRCM) to capture deep interrelation text information. We design a comparison sample generation method which can produce several instances for one input sentence and compare the distance between positive instance and negative ones, so as to improve the hidden text information mining ability. Experiments conducted on relation extraction common datasets confirmed the promotion of ZRCM compared with the existing methods. Especially, our model can improve the F1 value by up to 7% at best. When there are fewer unseen relations to predict, our model can achieve better performance.

Index Terms—relation extraction, zero-shot learning, contrastive learning

I. Introduction

Relation Extraction (RE) is a task of extracting the possible relation between a given ordered pair of entities and the relevant context information from one passage or sentence. Relation extraction is a predecessor task of NLP and plays an important role in many downstream tasks, such as constructing or expanding knowledge graph [15] and question answering system [8]. Existing methods for RE often require large-scale labeled datasets to conduct the training process, these labeled datasets require manual pre-process, which are time-consuming and labor-intensive [13]. One possible way to solve this problem is to use distant supervision generate annotation datasets [10]. However, existing datasets are always difficult to cover all the relations, due to the variety of relations. If the training datasets do not contain the labels for all relations, the existing supervised learning methods for relation extraction cannot handle this situation adequately.

It is unreasonable to assume that training data always contains relations which need to be distinguish in real world. Therefore, it is crucial to explore new models to predict new classes which are not defined or observed in advance, such tasks are called Zero-Shot learning (ZSL) [7]. By applying ZSL to relation extraction, Zero-Shot Relation Extraction (ZS-RE) become a paradigm which has a logical strategy in dealing with unseen relation. ZS-RE can identify new relations which don't have labeled data. In other words, it requires the model to predict whether an input sentence containing with two entities matches one relation whose description can be found from a series of relations. The core of ZS-RE is to find the indicative text information that can be used to judge whether the input sentence contains unseen relations, and the model can use it as a comparison basis to define the type of the relation among the input sentence. ZS-RE can also be regarded as a textual entailment task [17] in a broad sense, which is applied to identify new relations without corresponding labeled data for training, or requires the model to predicts whether the input sentence containing two entities also contains a relation matching the description of a given relation. The unseen relations are predicted through the existing labeled data, thus avoiding the over-dependence on the training data of a large number of sequence annotation tasks.

Studys on ZS-RE can be divided into two categories by the indicative text information of the comparison part. One is the generative method. The generative method means that the index text information required by ZSL need to be generated through the existing model. For example, [14] question the relations in the sentences using the generation method of Query in Q&A and take these questions as the key information for subsequent model training. Another approach we called self-labeling method uses explanatory text information in the network as indicator text information. Through the contrastive learning method [3], [4], the self-labeling method can achieve the purpose of training model by reducing the distance between the vector representation of input sentences generated by the representation learning method and the indicator text information. We call it self-labeling method because the text information indicator we need is usually the definition of a word or phrase, such as naturally marked and interpretable texts, and these texts are often highly reliable and relatively easy to obtain, such as Wiki-data. Owing to the advantages of the self-labeling

DOI reference number: 10.18293/SEKE2022-032

method, this paper adopts the relational description as the indicative text information of the contrastive learning, and identifies the unseen relations by extending the semantic space with generalization ability through the representation learning. But it is difficult to learn the semantic space with robustness and generalization, especially the over-fitting problem in ZSL and the uncertainty of unseen relation prediction.

In order to improve the performance of the model and get more generalization ability of the model and weaken the influence of the fitting problem, we propose an adversarial contrastive learning method for relation extraction, design a negative sample generator for zero-shot contrastive learning to generate different negative complements which can complete missing information or weaken the information that positive samples pay much attention to, so that the model can obtain better result in generalization ability. Finally, the model parameters are optimized by loss function specially designed to get better performance. This paper provides a method to improve the accuracy and stability of ZSL relation extraction by constructing a targeted contrastive learning method based on the comparative information of relation descriptions and BERT [6] pre-training language model. The contribution of this article can be summarized as follows:

- In general, we use relational description information to construct a contrastive training method for zero-shot relation extraction, which not only has the interpretability and simplicity of natural language, but also has significant advantages for the relation extraction task involving unseen relations.
- Methodologically, considering the generalization ability of the model, we construct a generation method of contrastive learning instances for relation extraction, and a negative samples generator for adversarial training, which lead to the significant improvement in the model accuracy, recall and F1 value.
- Experimentally, sufficient experiments based on two representative datasets demonstrate the effectiveness of our contrastive learning method for relation extraction and the effectiveness of the negative sample generation method in contrastive learning.

II. Problem Definition

We consult the definition of zero-shot relation extraction given by [1] and transform it to suit our model requires. There is a text collection W, which includes a number of sentences S, each sentence contains two entities e_1 and e_2 respectively. Besides, each sentence also contains a specific relation R with its description d for the entity pair e_1 and e_2. There is another text collection $\bar{W}$, together with a serious of sentences $\bar{S}$, each of which also contains an entity pair $\bar{e_1}$ and $\bar{e_2}$. But we do not know what the relation $\bar{R}$ between entities is. Our goal is to train a model M, which deduce the relation $\bar{R}$ in sentences $\bar{S}$ from training by input sentences S together with its relation description d. That is to say, we use model to infer unseen relation with labeled data. Out input can be expressed as $P = (S_0, e_1, e_2, R, d)$, through the model, the result can be indicated as $M(P) \Rightarrow \bar{R} \in \bar{S}$. That is the process of judging unseen relation by seen relation.

In the process of model training, in order to capture more detailed textual information, we put forward a Negative Sample Generator. It generates multiple different representation instances of the same sentence, these negative samples S_1, S_2, S_3 carry different levels of information containing the original data. We improve the model effect by reasonably adjusting the composition and training of examples to maximize the span between positive and negative samples. Then, our method is updated as: $M(\bar{P}) \Rightarrow \bar{R} \in \bar{S}, \bar{P} = (S_0, e_1, e_2, R, d)$, where $i = 0, 1, 2, 3$. $\bar{P}$ represents the new input to the model. This approach promotes our method to mine more detailed hierarchical information, as detailed in chapter 3.

Relevant researches on zero-shot relation extraction are few. This paper is similar to [1], who use training instances formed by input sentences and relation descriptions map into the embedding space by minimizing the distance between sentences and relation descriptions jointly and classify the seen and unseen relations. Since zero-shot relation extraction can be regarded as a text implication task, methods containing text implication thought can also achieve similar effects, like Enhanced Sequential Inference Model (ESIM) [2] and Conditioned Inference Model (CIM) [16]. By pairing each input sentence with each relation description, they trained the model to answer whether the paired text was contradictory or implicit. These models can infer and predict unseen relations by training input sentences and unseen relation descriptions. Contrastive Learning is often regarded as unsupervised learning or self-supervised learning method. Its core purpose is to obtain vector representations which are compatible with downstream work by limited data and labels. Thus, the use of labeled data and the choice of training methods are very important. In this paper, contrastive learning is included in the comparison of text-implied task methods, and the learning ability and generalization ability of model representation learning are improved by increasing the available information with samples generated.

III. The Proposed Model

In this section we discuss our model ZRCM whose basic process is shown in 1 The Negative Sample Generator and our training method are also explained in detail.

A. Model Process

We tokenize the sentences and send them into encoder to obtain contextual representation. For sentence $S_0^i, S_0^i \in S_0$, every token is represented as a vector H_i, where $i \in [0, n]$ and n denotes the number of tokens in S_0^i. [CLS] contain the whole information in sentence indicated as H_0,

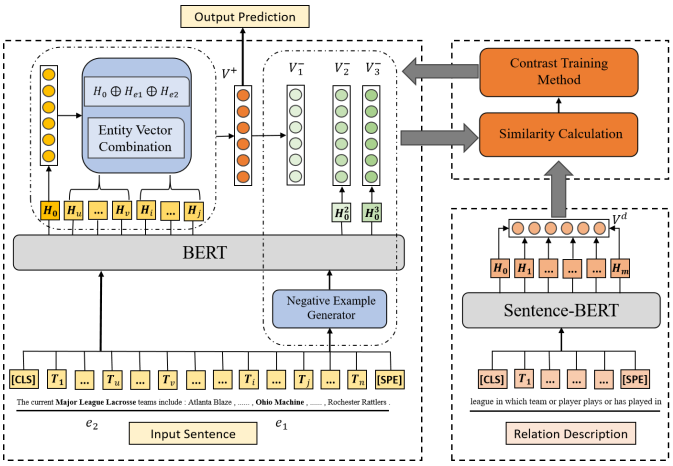


Fig. 1. The architecture of ZRCM.

where [CLS] is the classification token for sentences . We extract the representation of two entities and concatenate the vector representations respectively:

$$H_{e1} = \frac{1}{j-i+1}\sum_{a=i}^{j} H_a, H_a \in H_1^e, \tag{1}$$

$$H_{e2} = \frac{1}{v-u+1}\sum_{a=u}^{v} H_a, H_a \in H_2^e, \tag{2}$$

where i and j represent the start and end position of the tokens for e_1, u and v represent the start and end position of the tokens for e_2. $H_1^e = \{H_a | i \leq a \leq j\}, H_2^e = \{H_b | u \leq b \leq v\}$. The vector representing the entities are finally obtained by mean pooling, respectively. To capture the further information in entities, we concatenate H_{e1}, H_{e2} and H_0 by a fully connected layer and activation operation which are added to H_0: $\widetilde{V}^+ = H_0 + W(tanh([H_0 \bigoplus H_{e1} \bigoplus H_{e2}])) + b$, where W and b are the parameters that model needs to learn, $\bigoplus$ represents the vector concatenation. Then $\widetilde{V}^+$ is sent through a fully connected layer and activation to get V^+, which is one of the most important generate representation. It is clear that the entity pairs confirmed as the kernel of RE, but we do not want to pay too much attention to it to weaken other important information in the input sentence. Thus, we use residual network structure concatenate H_{e1}, H_{e2} and H_0 to construct V^+, which represents the relation representation contained in our input sentence for this model. On the other hand, the relation description d in input is feed to Sentence-BERT model, and which also be represented as a vector representation, donated as V^d. Noted that we need relation descriptions to modify our input representation, which are fixed and generated by the Sentence-BERT provided by [12]. The closer the distance $D(V^+, V^d)$ between V^+ and V^d, the more expressive V^+ the model obtain, where D represents a distance algorithm for calculating similarity of distance, including three possible choices of Euclidean distance, inner product and cosine similarity. In order to fully dig out the information of the potential implication relation contained of the input sentences we generate a negative sample generator for the zero-shot relation extraction. The detailed information of the negative sample generator will be introduced in chapter 3. We send input sentence S_0^i to negative sample generator and we will get three different processed sentences S_1^i, S_2^i, S_3^i, which will be feed into our model together with S_0^i in our input $\bar{P}$.

B. Negative Sample Generator

The choice of negative samples has a considerable influence on the effect of comparative learning. Negative samples are generally borne by other positive samples in batch. Although this method is simple and convenient, it obviously has certain shortcomings. From an empirical point of view, the negative samples selected in this way are highly random and may cause fluctuations in the training of the model, and the validity of negative samples produced in this way needs further confirmation. Such selections of negative samples are difficult to stand up in terms of interpretability. The false negatives generated by other sentences in the batch may have a similar relation description with the positive sample, which will greatly interfere with the training of the model. Therefore, we design a negative sample generator for zero-shot relation extraction. It can generate three different types of negative samples from the positive samples. We named them Random Negative Samples (RNS), Relational Negative Samples (ReNS), and Entity Negative Samples (ENS). What needs to be mentioned is that relational negative samples and entity negative samples are weak negative samples produced in order to cooperate with the contrastive learning of positive samples. They have a small gap with the positive samples and need to be used in conjunction.

For a positive sample of a trained sentence S_0^i, we take one vector representation from other sentences in the same batch which is the farthest away from the vector representation of this relation as the random negative sample S_1^i. In order to retain the real information and produce a large gap with the vector representation of the positive sample, we choose to use the same method as the positive sample for training and generate the corresponding vector representation V_1^-. Then we can express it as: $D(V_1^-, V^d) = max(D(V_i^+, V^d)), i \in b$, where b represents the size of the batch size, V_i^+ represents other sentences contained in the same batch, and D represents the similarity function. Although the selection of random negative samples is uncertainty, it can improve the generalization ability of the model through randomness in zero-shot learning scenarios.

In order to more deeply capture the relational information contained in the sentence, we have generated weak negative samples V_2^-, that is, relational negative samples.V_2^- is generated by mask tokens that may directly

indicate relational words in the sentence. We maximize the distance $D(V_2^-, V^d)$ by the sentence and the vector representation of the relation description V^d to obtain information which may be missed by the positive sample as a supplement. Many sentences may not contain tokens which can directly indicate relation words. When this happens, we use a certain percentage of tokens in the random mask sentence as an alternative method (the entity pair tokens are not included).

For zero-shot tasks, over-fitting is one of the most serious problems. We think this is a major problem that limits its generalization performance for this model that pays too much attention to entity information. Therefore, we mask the tokens of the entity pair in the sentence to generate the negative input of the entity, and maximize the distance $D(V_3^-, V^d)$ between the generated negative vector representation V_3^- and the relation representation V^d.

For the last two methods, they generate weak negative samples and have no need to focus on the entities. We directly use the hidden layer corresponding to [CLS] to pass through an activation function layer and a dropout layer as V_2^- and V_3^-, respectively. In this way, through the negative sample generator, a positive sample generates several negative samples with different emphasis information.

C. Training

The training of ZRCM consists of two objectives. Firstly, by generating suitable positive and negative samples, and increasing the span between positive and negative samples as much as possible to obtain a more generalized model effect. We compare the distance of the positive case with the three negative case distances as a pair of calculations:

$$C(D(V^+, V^d), D(V_i^-, V^d))$$

$$= max(0, \gamma - D(V^+, V^d) + D(V_i^-, V^d)), \tag{3}$$

where $i = 1, 2, 3$. Then the goal of our model is expressed as:

$$\mathcal{C} = \sum_{i}^{1,2,3} [C(D(V^+, V^d), D(V_i^-, V^d))] \tag{4}$$

where γ is a hyper-parameter, whose purpose is to keep a certain buffer space for the distance difference between the positive sample and the negative sample. In the training, we iterate the computation to make $D(V^+, V^d)$ obtain a larger value while ensuring that $D(V_i^-, V^d)$ is smaller, that is we increase the distance between positive samples and the relation discription representations while reduce the distance between negative samples and the relation discription representations.

Our second objective is to use cross-entropy loss to maximize the accuracy of Relation Label Classification based on visible relations:

$$\mathcal{D} = max(0, \sum_{j}^{0,2,3} (-1)^{2\times j} V_j^R log(\bar{V_j^R}) + \beta), \tag{5}$$

where V_j^R represents the visible relation, $\bar{V_j^R}$ represent its relation represents the probability distribution of the corresponding visible relation prediction, and β is a hyper-parameter, in order to ensure the full use of the training data. In addition, we also generated the corresponding visible relation prediction distributions for the other negative samples which are divided and can ensure the sufficiency of the negative samples we generate when used in the first objective. Because the syntax structure of the input two negative samples is very similar to the positive sample, and it is necessary to ensure the difference in this way. The larger the relation prediction gaps are, the more representative the negative samples we generated and the more helpful for generalization ability. In general, the larger the $\mathcal{D}$, the higher the probability that the predictions are correct.

Combining the two objectives described above, our final objective function can be expressed as:

$$L = (1 - \alpha) \times \mathcal{C} - \alpha \times \mathcal{D} \tag{6}$$

where $\mathcal{C}$ comes from Eq.(4), and $\mathcal{D}$ comes from Eq.(5), which are the hyper-parameters. All the hyper-parameters mentioned in the model will be studied and discussed in detail in the subsequent experimental part.

IV. Experiments

A. Experimental Setup

Datasets. We use two datasets for experiments, FewRel [9] and Wiki-ZSL [5]. FewRel has 70,000 sentences selected by a large number of crowd workers from Wikipedia, which contains about 100 relations. Then use the distant supervision method to complete the preliminary labeling, and then manually filter out the wrong sentences, so that the dataset becomes a clean RC dataset, which has 56,000 examples containing 80 different relationships. For another data set Wiki-ZSL, which originally came from Wiki-KB, it was also generated with a distant supervision method, with 93483 examples and 113 different relations. The two data sets have one thing in common, that is, they are both built on the basis of data in Wikipedia, which allows them to be accurately linked to the Wiki-data knowledge base. This provides possibility and great convenience for the zero-shot relation extraction.

ZSL Experimental Setup. We divide a data set into three different predictive relation quantities $\mathfrak{a}$. That is, one part of the data set is used for training, and the other part is used for prediction. $\mathfrak{a}$ has three possible options: 5, 10, and 15. In order to meet the requirements of zero-shot learning, it is necessary to ensure that there

is no intersection between the trained relational data and the predicted relational data. We use Precision, Recall and F1 value as a measurement method to evaluate the effect of the experiment. We repeat the experiment for more than 5 times, randomly select relations as test set, make the rest of them as training set, report the best result of every single experiment and evaluate the final results comprehensively. We do our best to ensure the comparability of our experimental method and other comparison methods.

Comparison Methods. R-BERT is a supervised method of relation extraction. It has excellent results in fully supervised relation extraction experiments but performs poorly on zero-shot prediction tasks. ESIM [2] and CIM [16] are two texts which contain tasks. They accept sentences and relation descriptions as input, and output a binary label indicating whether they match semantically. ZS-BERT is the baseline of this experiment, and it has an experimental effect far superior to other models by using zero-shot learning. The experimental results of other comparison methods are from [1]. In general, we hope to show the advantages or disadvantages of this method by comparing the results with other methods.

Parameter Settings. Our model is based on Hugging Face and PyTorch. We use the Adam [11] optimizer, the batch size is set to 4, the size of the hidden layer is 768, the embedding dimensions of the input sentences and the dimensions of the attribute vectors are 1024. To make the experiments easy to compare, we used exactly the same data set with the traditional evaluation indicators: Precision, Recall and F1. For different datasets our hyperparameters are not the same. For FewRel, $\alpha = 0.4, \beta = 4, \gamma = 7.5$, for Wiki-ZSL, $\alpha = 0.4, \beta = 0, \gamma = 7.5$. α is the weight parameter of the balances loss function. The similarity function D has been compared through many experiments and found that using inner product will achieve a more stable and high-quality effect.

TABLE I
The comparative results of the experiments

	Wiki-ZSL			FewRel		
	$\mathfrak{a} = 5$			$\mathfrak{a} = 5$		
	P	R	F1	P	R	F1
R-BERT	39.22	43.27	41.15	42.19	48.61	45.17
ESIM	48.58	47.74	48.16	56.27	58.44	57.33
CIM	49.63	48.81	49.22	58.05	61.92	59.92
ZS-BERT	71.54	72.39	71.96	76.96	78.86	77.90
ZRCM	76.15	77.1	76.6	86.70	84.51	85.60
	$\mathfrak{a} = 10$			$\mathfrak{a} = 10$		
	P	R	F1	P	R	F1
R-BERT	26.18	29.69	27.82	25.52	33.02	28.20
ESIM	44.12	45.46	44.78	42.89	44.17	43.52
CIM	46.54	47.90	45.57	47.39	49.11	48.23
ZS-BERT	60.51	60.98	60.74	56.92	57.59	57.25
ZRCM	62.41	64.16	63.27	53.67	53.96	53.81
	$\mathfrak{a} = 15$			$\mathfrak{a} = 15$		
	P	R	F1	P	R	F1
R-BERT	17.31	18.82	18.03	16.95	19.37	18.08
ESIM	27.31	29.62	28.42	29.15	31.59	30.32
CIM	29.17	30.58	29.86	31.83	33.06	32.43
ZS-BERT	34.12	34.38	34.25	35.54	38.19	36.82
ZRCM	33.47	36.71	35.01	40.27	40.72	40.50

B. Experimental analysis

Main Experiment. We predict the experimental results of different numbers of unseen relations which can be seen from Table 1. First of all, we can see that our model's effect is significantly better than other models, especially when $\mathfrak{a} = 5$. Our model outperforms the second model about 7.7% at F1 value in FewRel dataset which can show the ability of case comparison method to capture potential information. For $\mathfrak{a} = 15$ our model can also achieve F1 value increase of about 3% at best, which reflects our model's superiority in predicting more generalization ability of unseen relations. When $\mathfrak{a} = 10$ ZRCM lags ZS-BERT by about 3.4% in FewRel, we believe that it may be due to insufficient data that the negative samples and positive samples are not reasonably divided. On the whole, when $\mathfrak{a}$ is smaller, the predicted unseen relations are fewer and the experimental precision recall and F1 value will be ignificantly higher than when $\mathfrak{a}$ is larger. This is predictable. On the one hand, we only need to predict fewer target relations, and at the same time, the sources of information we can obtain are also increasing. It can be seen that although the textual implication models such as ESIM and CIM have a higher improvement than R-BERT, there is a big gap between ZS-BERT and this experiment, which shows that the textual implication tasks cannot be perfectly covered and fit unseen. For ZS-BERT and ZRCM, it can be seen that the results of our model have substantial improvements on ZS-BERT, which reflects the effectiveness and superiority of the overall process setting of our model.

Ablation. In order to fully demonstrate the effectiveness of our negative samples, we design the ablation experiments for them based on the method of controlling variables. That is to say, we eliminate the distance constraint between the label and the input hidden layer vector and only consider the unilateral impact of negative samples on the experimental result. ZRCM is our model, ZRCM* expresses our first goal, that is, the result of ZRCM after removing the Relation Label Classification. RNS, ReNS and ENS are the samples that we build with Negative Sample Generator to compare with the positive samples. It can be obtained from the data analysis in *Table*2 that ZRCM get the better results than ZRCM*, which proves

the contribution of our goal 1 to the overall model effect. Besides, for three comparative negative samples we can see that RNE contributes the most to the result which is understandable, because although other comparative negative samples may carry more deep attributes, they still need a sufficient distance to optimize the model. Since different data sets have different text features, it is reasonable that the results of the same comparison negative sample on the two data sets are different.

TABLE II
The impact of different negative sampling methods

F1	FewRel	Wiki-ZSL
ZRCM*	0.4387	0.2950
ZRCM* - RNS	0.1314	0.0748
ZRCM* - ReNS	0.4178	0.3236
ZRCM* - ENS	0.4591	0.3098
ZRCM	0.4544	0.3298

Hyperparameter Experiment. In order to achieve the optimal performance of our model, we conduct extensive experiments to judge the effect of different hyperparameters on the performance.

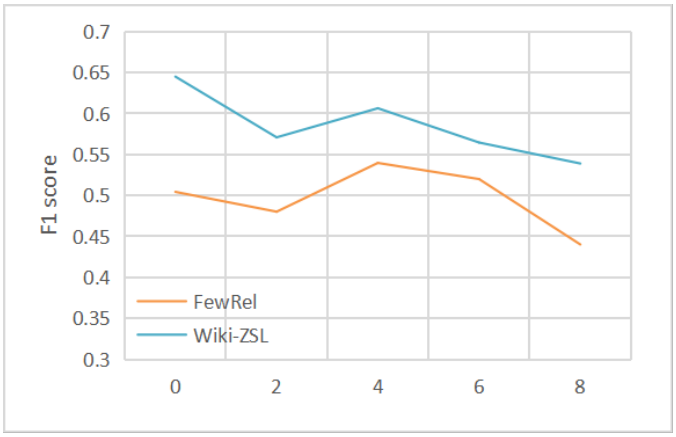


Fig. 2. Changes in F1 under the influence of β

In particular, we show how changes in the parameter β in Eq.(5) affect the model performance, β is a boundary parameter. To determine that the objective 2 of the experiment obtains a more ideal optimization distance, we tried different values of β when $\mathfrak{a} = 10$. For generating better negative samples, objective 2 is designed to help the model generate more representative negative representations, which makes the vector representation of the labels have greater distance from the vector representation of the negative samples generated by the negative sample generator. The results on two datasets are exhibited on Fig. 2, the two curves have the same trend, but they achieve the best performance in different place. It's reasonable for the inherent differences in the textual information of the two datasets.

V. Conclusion

In this work, we present a novel method matching representation learning for Zero-Shot Relation Extraction. With the Negative Sample Generator, our model can capture depth information for the input sentences. Besides, we use multi-task learning structure with negative sample training model. Results show that our model can substantially improve the performance, we also carry out extensive experiments which can verify the effectiveness of the designed adversarial training. The ability to understand and summarize the text and the problem of over-fitting are the important factors which limit the effectiveness of the model, which also forms the basic idea of our method.

Acknowledgment

This research is supported by the National Key Research and Development Program of China (Grant No. 2019YFB1706101), Natural Science Foundation of Chongqing, China (No. cstc2020jcyj-msxmX0900), and the Funda-mental Research Funds for the Central Universities (Project No. 2020CDJ-LHZZ-040)

References

[1] Chih-Yao Chen and Cheng-Te Li. 2021. ZS-BERT: Towards Zero-Shot Relation Extraction with Attribute Representation Learning. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Pages: 3470–3479. Association for Computational Linguistics.

[2] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1657–1668 Vancouver, Canada. Association for Computational Linguistics.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119:1597-1607.

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, Kaiming He. 2020. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297.

[5] Sorokin Daniil, and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1784-1789 Copenhagen, Denmark. Association for Computational Linguistics.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171-4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[7] Kodirov Elyor, Tao Xiang, and Shaogang Gong. 2017. Semantic autoencoder for zero-shot learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3174-3183.

[8] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, Jiwei Li. 2019. Entity-Relation Extraction as Multi-Turn Question Answering. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1340-1350 Florence, Italy. Association for Computational Linguistics.

[9] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, Maosong Sun. 2018. FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4803-4809 Brussels, Belgium. Association for Computational Linguistics.

[10] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, page 3060–3066. AAAI Press.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980.

[12] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982-3992, Hong Kong, China. Association for Computational Linguistics.

[13] Yong Shi, Yang Xiao, Pei Quan, Minglong Lei, Lingfeng Niu. 2021. Distant Supervision Relation Extraction via Adaptive dependency-path and Additional Knowledge Graph Supervision. Neural Networks, 134:42-53.

[14] Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, Eneko Agirre. 2021. Label Verbalization and Entailment for Effective Zero and Few-Shot Relation Extraction. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1199–1212. Association for Computational Linguistics.

[15] Haoze Yu, Haisheng Li, Dianhui Mao & Qiang Cai. 2020. A relationship extraction method for domain knowledge graph construction. World Wide Web 23.2 (2020): 735-753.

[16] Shu Zhang, Dequan Zheng, Xinchen Hu and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. Proceedings of the 29th Pacific Asia conference on language, information and computation, pages 73-78 Shanghai, China.

[17] Obamuyide, Abiola, and Andreas Vlachos. 2018. Zero-shot relation classification as textual entailment. Proceedings of the First Workshop on Fact Extraction and VERification (FEVER).

Similarity matching of time series based on key point alignment dynamic time warping

Yangzheng Li, Zhigang Chen+, Xiaoheng Deng
School Of Computer Science And Engineering
Central South University
Changsha, China
+ Corresponding author mail: czg@csu.edu.cn

***Abstract*—Similarity measurement is an important basis in time series analysis. Among them, dynamic time warping distance (DTW) is considered to be the most effective distance measurement method. However DTW's huge computational overhead is difficult to meet the application requirements in the era of big data. Previous optimization methods often focus on reducing unnecessary calculation objects and do not involve warping distance calculation itself. After studying many related optimization algorithms, we propose a DTW matching algorithm based on key structure point alignment. By extracting the key structure points of time series and calculating the warping alignment relationship between the key structure points, the constraint range of the cumulative distance matrix of the approximate optimal warping distance from the path is mapped, which greatly reduces the amount of calculation of the distance cumulative matrix, then approximate warping distance can be calculated quickly. The experimental results show that the calculation speed of our method is significantly improved compared with the traditional algorithm in similarity matching, and it also has a good performance in classification accuracy.**

Keywords-time series; data mining; dynamic time warping distance

I. INTRODUCTION

Time series is a concept derived from data mining, which generally refers to an ordered set of the same statistical index values arranged according to their time order. The analysis of time series has made important applications in the fields of finance, medical treatment, meteorology, geology and so on[1-4].

The similarity measurement is an important basis of time series analysis. Comparing and judging the similarities and differences of two groups of time series can further realize the classification and clustering of time series, thus it can be used as an important basis for time series analysis. In related research, distance is usually used as the measure of similarity between time series. The smaller the distance is, the more similar the time series is. Euclidean distance is the most classical time series distance measurement, which is simple and fast to calculate, but the calculation method of point-to-point alignment can not be used for the comparison of unequal time series, nor can it solve the problems of distortion, scaling and drift on the time axis, so it is rarely used in practical application.

Another common classical distance measure, dynamic time warping (DTW)[5], has been proposed for many years and still plays an important role. It aligns each series point through dynamic programming to find the minimum cumulative distance, allows warping alignment in time axis, and solves the limitation of Euclidean distance, but the calculation cost of cumulative distance matrix is large, The time complexity of DTW is $O(n^2)$, which is hard to apply to the time series analysis of big data.

Many other distance measurement algorithms have also been proposed. For example, the symbolic editing distance based on time series (EDR)[6], the longest common subsequence (LCSS)[7], the most similar subsequence (TSW)[8], etc. they are also algorithms based on dynamic programming, which has considerable time complexity compared with DTW. In addition, there are non dynamic programming methods, although the complexity is lower, but the accuracy is often insufficient. The maximum shifting correlation distance[9] has only the time complexity of $O(n)$. It finds the maximum Pearson correlation coefficient of the two time series through the sliding window to obtain the approximate distance. The fragment alignment distance[10] uses the approximate derivative and the number of continuous segments of each segment of the sequence to represent a segment of subsequence, and calculates the distance in the way of approximate diagonal alignment. Since the complex alignment is not considered, the computational complexity is reduced to linear. The fluctuation features distance[11] considers the trend change of time series and calculates the distance by weighting the change value. The MPdist[12] uses the method of matrix representation of the sequence, divides the sequence into multiple subsequence groups, puts forward the closest subsequence pair to form a sequence, and takes a large enough value as the distance result.

However, due to its excellent universality and matching accuracy, DTW is still difficult to replace. In order to solve the computational cost of DTW, researchers have proposed many methods, such as the following boundary distance[13,14], early

This work is supported by the Intelligent software and hardware system of medical process assistant and its application belong to "2030 Innovation Megaprojects" - New Generation Artificial Intelligence (Project no. 2020AAA0109605), and Major special project of Changsha science and technology plan (Project no. kh2103016).
DOI reference number: 10.18293/SEKE2022-021

abandonment[15]. At the ACM SIGKDD conference, Thanawin et al.[16] proposed the UCR suite, which integrates important DTW acceleration methods in the past. These methods try to skip the calculation of part of the warping distance, so as to save the overall calculation time, and do not involve the calculation of the warping distance itself. Another idea is to reduce the dimension of time series, and then calculate the distance by dynamic programming as usual. Pr-DTW[17] method represents the segmented time series by weighting multiple statistical indicators, and calculates the distance after greatly reducing the amount of data. Soft-DTW[18] proposes a method to find the approximate warping path by subdividing the warping distance matrix step by step, which does not need to calculate the complete warping distance matrix.

Based on these studies, this paper proposes a distance measurement algorithm based on key point alignment. The cumulative distance is calculated by finding the near optimal path through the key points, which significantly reduces the computational complexity. The algorithm in this paper is used for the measurement of 1-NN classifier for experimental test. The results show that compared with the traditional algorithm, the algorithm in this paper significantly reduces the time cost and maintains good matching accuracy.

II. PRE KNOWLEDGE

A. Classical dynamic time warping method

Set the time series as $X\{x_1,\ x_2,\ x_3, \dots \dots x_m\}$, and $Y\{y_1,\ y_2,\ y_3,\ \dots \dots y_n\}$, define the DTW distance between the two time series as *DTW* (*X*, *Y*), and construct an matrix $\boldsymbol{D}$ of size $m \times n$, calculate the value of each matrix element $\boldsymbol{D}[i][j]$ ($i \in [1, m], j \in [1, n]$) according to the following method:

$$\boldsymbol{D}[1][1] = d(1,1) \tag{1}$$

$$\boldsymbol{D}[i][j] = d(i,j) + \min \begin{cases} \boldsymbol{D}[i-1][j] \\ \boldsymbol{D}[i][j-1] \\ \boldsymbol{D}[i-1][j-1] \end{cases} \tag{2}$$

$$d(i,j) = \left|x_i - y_j\right| \text{ or } (x_i - y_j)^2 \tag{3}$$

Then $\boldsymbol{D}[m][n]$ is *DTW* (*X*, *Y*). In the process of calculating *DTW* (*X*, *Y*), connect the minimum values of $\boldsymbol{D}[i][j]$ to obtain a path of warping distance accumulation:

$$path = \{(1,1), \dots \dots (i,j) \dots \dots, (m,n)\} \tag{4}$$

where the two values in each binary represent two elements from two time series, and *i* and *j* represent their time axis positions in their respective series. When *path* can make the cumulative distance function $\sum_{i,j\ in\ path}^{len(path)} d(x_i, y_j)$ gets the minimum value, that is, *DTW* (*X*, *Y*), at this time, the *path* is called the optimal alignment path of these two time series, and each pair of binary in the *path* is called the "alignment" relationship.

B. Constraint range

The calculation of *DTW* (*X*, *Y*) needs to traverse all elements in the calculation matrix. Its complexity is $O(m \times n)$, can be recorded as $O(n^2)$. Global constraint is an idea of optimizing DTW distance calculation, which reduces the amount of calculation by limiting the cumulative range of warping distance. For $X\{x_1,\ x_2,\ x_3, \dots \dots x_m\}$ and $Y\{y_1,\ y_2,\ y_3,\ \dots \dots y_n\}$, the element that far from the diagonal in the warping distance matrix, such as $\boldsymbol{D}[1][n]$, reflect the alignment between the first element in *X* and the last element in *Y*. This means that time series is aligned with the time axis in a very distorted state.

This alignment usually does not conform to the actual situation. The value on the matrix is often too large to be added to the final result. In fact, this part of the calculation can be omitted. In the methods of Itakura constraint[19] and Sakoe-Chuba constraint[20], as shown in Figure 1, the calculation range is limited near the diagonal of the warping distance matrix, as the dark part in the figure.

However, this global constraint method lacks flexibility. If the alignment path is outside the constraint range, there will be a large error between the calculation result and the optimal distance. When the alignment path is in the constraint range, there is still a large computational overhead.

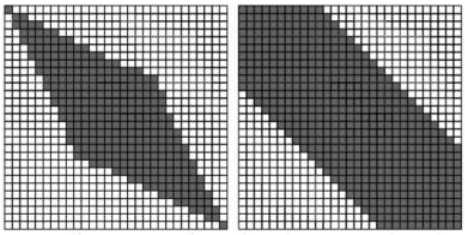


Figure 1. Itakura constraint and Sakoe-Chuba constraint

III. METHOD

In fact, to get the final DTW distance value, only needs to calculated the value on the optimal alignment path in the matrix. Because the value on the path only depends on the minimum of the three candidate cumulative distance values. The problem is that the optimal alignment path cannot be known until the complete warping matrix is calculated, but the near optimal alignment path can be found through some methods. By constraining the calculation range of the waping distance matrix near the near optimal path, a very close DTW distance can be obtained.

A. Find key structural points

Due to the continuity of the time series, a few key structural points in the time series can reflect the approximate trend shape of the whole time series image. Therefore, a set $X'\{(k_1, x_{k1}),\ (k_2, x_{k2}), (k_3, x_{k3}), \dots \dots (k_p, x_{kp})\}$ can be used to represent time series $X\{x_1,\ x_2,\ x_3, \dots \dots x_m\}$ approximantely. Where k_i represents the time axis position of each key structure point, *p* is the number of key stuctural points in the time series image, and *p* is much smaller than *m*.

In order to find the key points of the time series, firstly use the PAA[21] method to process the time series data to reduce the noise jitter of the time series image and improve the operation efficiency. Then the extreme points in all time series image points should be screened. The left and right derivatives of the extreme points are opposite, as shown follows:

$$\{x_i | (x_i - x_{i-1})(x_{i+1} - x_i) < 0\} \quad (5)$$

And add those non extreme points x_i with large turning points, as shown follows:

$$\{x_i || arctan(x_i - x_{i-1}) - arctan(x_{i+1} - x_i)| > \gamma\} \quad (6)$$

Where γ is the threshold, its value is π/6 in this paper.

Some time series data may have local jitter, resulting in the aggregation of key points in a small section. In order to ensure that the key points reflect the overall morphological trend of the time series, these points need to be further filtered to reduce local aggregation. When the distance between a newly added key point and the previous key point is lower than the threshold γ_c, it will not be added to the key points set. And γ_c can be calculated as equation (7):

$$\gamma_c = 0.1 * \sqrt{(\max(x) - \min(x))^2 + m^2} \quad (7)$$

In addition, the first and last points of each time series are specified as key points.

Assuming that there are time series $X\{x_1, x_2, x_3, \ldots\ldots x_m\}$ and $Y\{y_1, y_2, y_3, \ldots\ldots y_n\}$, $X'\{(k_1, x_{k1}), (k_2, x_{k2}), \ldots\ldots, (k_i, x_{ki}), \ldots\ldots (k_p, x_{kp})\}$ and $Y'\{(l_1, y_{l1}), (l_2, y_{l2}), \ldots\ldots, (l_j, y_{lj}), \ldots\ldots (l_q, y_{lq})\}$ are obtained after key point filtering. Where k_i and l_j respectively represent the time axis position of key points in the corresponding time series, and p and q are the number of key points in the corresponding time series.

B. Key structural points alignment

Elements of the distance accumulation matrix $\boldsymbol{K}$ of the key point sequence X' and Y' is calculated as follows, and the size is $p \times q$:

$$\boldsymbol{K}[1][1] = dk(1,1) \quad (8)$$

$$\boldsymbol{K}[i][j] = dk(i,j) + \min \begin{cases} \boldsymbol{K}[i-1][j] \\ \boldsymbol{K}[i][j-1] \\ \boldsymbol{K}[i-1][j-1] \end{cases} \quad (9)$$

$$dk(i,j) = |x_{ki} - y_{lj}| * (1 + |k_i/k_p - l_j/l_q|) \quad (10)$$

When calculating the distance accumulation matrix of key points, in addition to calculating the difference of time series elements corresponding to key points, it can be multiplied by the correction value of time axis position to obtain a more reasonable alignment relationship. After calculating the cumulative distance matrix $\boldsymbol{K}$ of key points, the optimal alignment path of key points $path_{kp}$ can be obtained.

We get the $path_{kp}$ reflecting the alignment relationship of key points. Due to the morphological continuity of time series, the alignment relationship between key points will be affected by key points. We can approximate the alignment path of the original time series by mapping the alignment path of the key points.

C. Mapping to original distance matrix

Construct the cumulative distance matrix $\boldsymbol{D}$ of the original time series, with the size of $m \times n$. The corresponding points in the $path_{kp}$ obtained in the previous step are mapped into the warping distance matrix of the original time series.

For each (i, j) in $path_{kp}$ indicates that the key point (k_i, x_{ki}) from X' is aligned with (l_j, y_{lj}) from Y', and mapped to the original distance matrix $\boldsymbol{D}$ is $[k_i][l_j]$. Each group of alignment relations in the $path_{kp}$ are mapped to $\boldsymbol{D}$ in turn to obtain a series of points on the near optimal path of the original time series, as shown in Figure 2.

Connecting them, as shown in the red part of the matrix in Figure 3, we can get a path similar to the optimal alignment path (the gray part of the matrix). As shown in Figure 4, expand the connection path outward by r areas range. The greater the value of r, the closer the result to the optimal distance can be obtained. In this paper, r = 1, and the final distance calculation constraint range can be obtained.

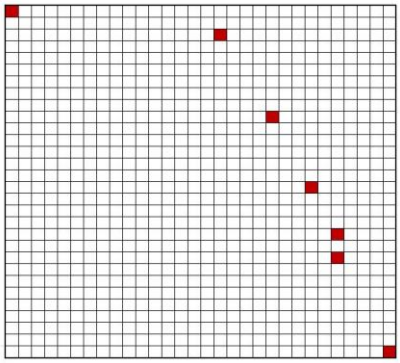


Figure 2. Mapping to distance matrix

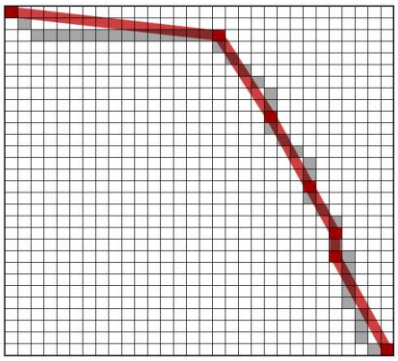


Figure 3. Approximate alignment path

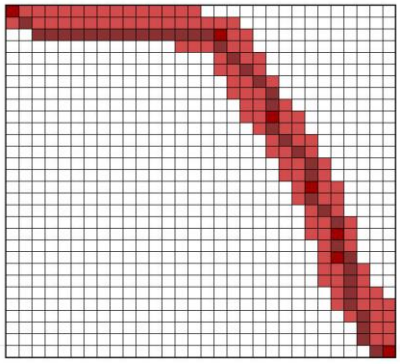


Figure 4. Constraint range

D. Calculate distance under the constraint range

Through the above methods, we get a constraint range close to the optimal warping path, which is based on the key structure alignment of the time series. Compared with the global constraint, it is more suitable for the actual alignment of the time series.

Under the constraint range, the calculation of distance matrix can be reduced, the result is closer to the optimal distance, and no need to traverse the entire distance matrix, but only the part of the matrix within the constraint range. The pseudo code for calculating the near optimal warping distance under the constraint range is as follows:

Algorithm 1 Constrainted warping distance

Input: Constraint *C* , series *X* and *Y*
Output: Distance

```
dist(1, 1) = d(1,1)
for (i, j) in C
  if (i-1, j) in C
        d1 = d(i-1, j)
    if (i, j-1) in C
        d2 = d(i, j-1)
    if (i-1, j-1) in C
        d3 = d(i-1, j-1)
  dist(i, j) = min(d1, d2, d3)
return dist(C.end)
```

IV. EXPERIMENTAL

UCR[22] is a representative data set in time series research. It contains 128 sub data sets from different sources, and the amount of data is very abundant. So we chose to run the experiment on the UCR dataset. In the experiment we take the measurement algorithm in this paper as the distance measurement of 1-NN classifier, preprocess it with PAA reduction method and Z-score standardization, and test its classification performance. We mainly investigate its running time and classification error rate, reflecting the calculation efficiency and matching accuracy of the algorithm respectively. We selected four algorithms DTW, EDR, LCSS, and TSW as the control. TABLE I lists the basic information of the data sets used.

In PAA reduction, we can control the processed time series length by modifying the window *w* size of PAA. The larger *w* is, the smaller the processed length is. When $w = 1$, the original time series is returned. We counted the time taken by each algorithm to complete a similarity search on the arrowhead training set for the same data set. Figure 5 shows the change of each algorithm time with the PAA window *w*. Among them, KPDTW is the algorithm of this paper, and the rest are four control algorithms. In the chart, as the *w* value increases, that is, the time series length decreases. We can see the time of the control algorithm decreases sharply, and the time of the algorithm in this paper decreases gently, which is lower than that of the control algorithm. Conversely, as the *w* value decreases, that is, the time series length increases, the time of the control algorithm increases sharply, while the time growth of the algorithm in this paper is relatively flat. This shows that the algorithm in this paper has better time complexity performance when the amount of data increases.

TABLE I. DATA SET INFORMATION

Name of data set	Size of train sets	Size of test sets	length	Number of classes
Adiac	390	391	176	37
ArrowHead	36	175	251	3
BME	30	150	128	3
CBF	30	900	128	3
Coffee	28	28	286	2
CricketX	390	390	300	12
ECG200	100	100	93	2
FaceAll	560	1690	131	14
GestureMidAirD1	208	130	Vary	26
GesturePebbleZ1	132	172	vary	6
Gunpoint	50	150	150	2
Lightning7	70	73	319	7
OliveOil	30	30	570	4
Plane	105	105	144	7
ShapesAll	600	600	512	60
Symbols	25	995	398	6
ToeSegmentation1	40	228	277	2
Trace	100	100	275	4
TwoPatterns	1000	4000	128	4
UMD	36	144	150	3
UWaveGestureLibraryX	896	3582	315	8
Wafer	1000	6164	152	2
WordSynonyms	267	638	270	25
Yoga	300	3000	426	2

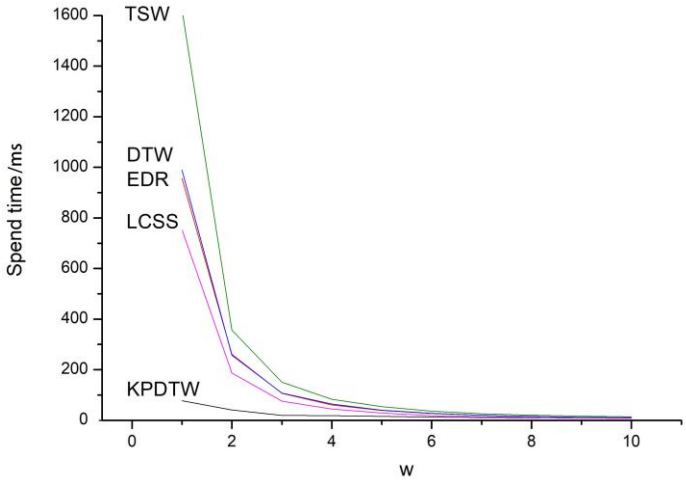


Figure 5. The relationship between spend time of each algorithm and *w* (ArrowHead dataset)

By running the classification test under 24 data sets, Figure 6 shows the cumulative running time of each algorithm. It can be seen that with the increase of the running data set, the cumulative time growth of the control algorithm is significantly higher than that of the algorithm in this paper which is expressed in KPDTW, especially in the data set with longer length, the running time will rise sharply. After running the classification tasks of 24 data sets, the algorithm in this paper saves several times the time overhead compared with the control algorithm.

In TABLE II, the classification error rates of different data sets of this algorithm and four control algorithms under 1-NN classifier are listed. The lower the error rate, the more accurate the matching result is. The algorithm with the best performance will be expressed in bold. The results show that the algorithm in this paper has excellent performance in five algorithms, achieves low error rate in 24 data sets, and the average ranking is slightly better than DTW algorithm.

Figure 7 shows the error rate comparison between the algorithm KPDTW in this paper and the control algorithm. It can be seen that the algorithm in this paper is highly close to DTW.

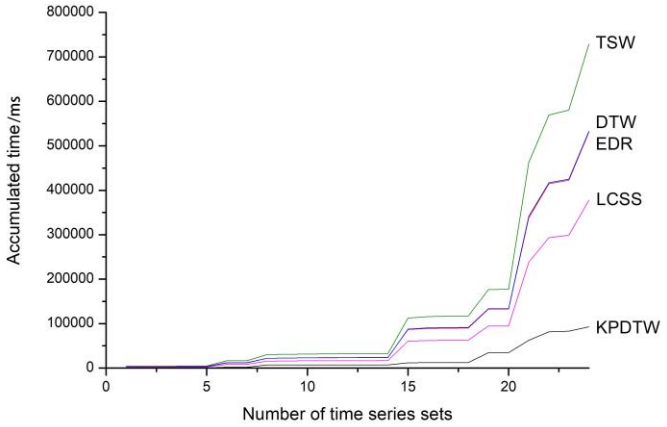


Figure 6. The accumulated time of each algorithms

TABLE II. CLASSIFICATION ERROR RATE OF ALGORITHM

Name of data set	Error rate				
	KPDTW	DTW	EDR	LCSS	TSW
Adiac	0.399	**0.391**	0.859	0.849	0.847
ArrowHead	0.251	**0.210**	0.268	0.229	0.234
BME	0.060	**0.053**	0.167	0.193	0.227
CBF	**0.002**	0.004	0.009	0.013	0.007
Coffee	0.036	**0.000**	0.071	0.071	0.071
CricketX	**0.267**	0.297	0.464	0.428	0.490
ECG200	**0.120**	0.150	0.310	0.170	0.140
FaceAll	**0.015**	0.025	0.050	0.045	0.090
GestureMidAirD1	**0.346**	0.362	0.477	0.777	0.685
GesturePebbleZ1	0.181	**0.175**	0.199	0.649	0.333
Gunpoint	**0.067**	0.087	0.160	0.080	0.093
Lightning7	**0.288**	**0.288**	0.341	0.301	0.301
OliveOil	0.167	**0.133**	0.833	0.833	0.833
Plane	**0.000**	**0.000**	0.028	**0.000**	**0.000**
ShapesAll	0.145	0.198	0.230	0.130	**0.100**
Symbols	**0.053**	0.062	0.179	0.104	0.124
ToeSegmentation1	**0.145**	0.170	0.195	0.215	**0.145**
Trace	**0.010**	**0.010**	0.030	0.230	0.110
TwoPatterns	**0.000**	0.002	0.050	0.090	0.095
UMD	**0.028**	**0.028**	0.139	0.250	0.243
UWaveGestureLibrary X	0.297	**0.227**	0.343	0.300	0.350
Wafer	0.005	0.005	0.055	**0.000**	0.005
WordSynonyms	0.280	**0.262**	0.455	0.340	0.345
Yoga	0.188	**0.156**	0.274	0.178	0.188
Average ranking	**1.542**	1.583	4.167	3.375	3.417

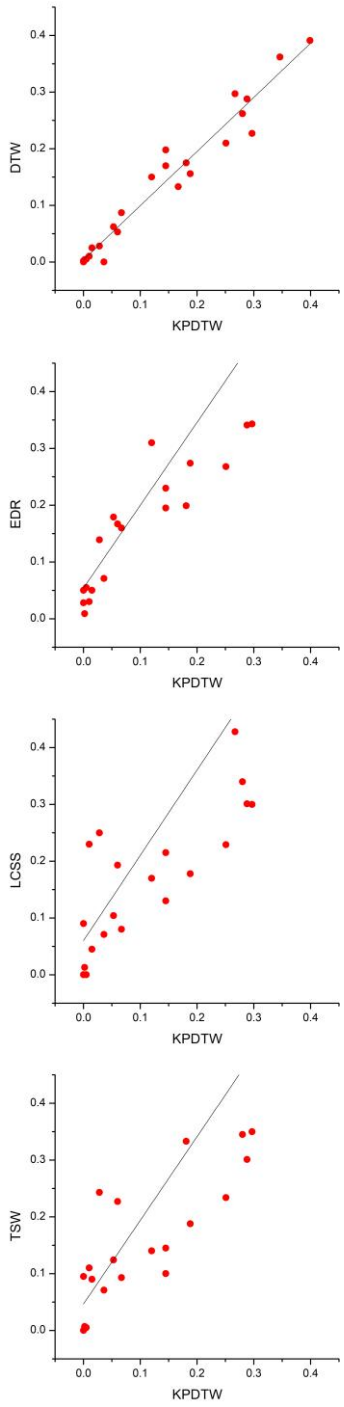


Figure.7 Error rate comparison of algorithms

V. RESULT ANALYSIS

As the most robust distance measurement method, DTW can ignore the distortion and stretching of time series in time axis through warping alignment, which has a good performance in many data sets. The algorithm in this paper is based on DTW and obtains the approximate optimal alignment relationship through the alignment of key points. The better the key point series fits the original time series, the closer the approximate

distance is to the DTW distance, and the matching accuracy of the experimental results is also similar to DTW. Although the algorithm in this paper obtains the approximate distance, the approximate distance also reduces the influence of local noise, it has achieved better results in the matching accuracy under some data sets. Compared with DTW, the algorithm in this paper only sacrifices little accuracy, but the computational efficiency is significantly improved.

Let the length of the time series be n, the time complexity of extracting the time series of key points be $O(n)$, the number of key points be p, the time complexity of key point alignment and near optimal path generation be $O(p^2)$, the constraint range obtained is a fixed width, and its coverage is only linearly related to the data length. So the time complexity of calculating the near optimal distance under the constraint is $O(n)$, The final total time complexity of calculating the distance once is O ($p^2 + n$).

In the worst case, that is, each point of the time series sequence is taken as the key point, it becomes the same time complexity O (n^2) as DTW algorithm. Generally, considering that the number of key structure points is often much less than the length of the whole time series, generally $p << n$, the complexity of this algorithm is significantly lower than $O(n^2)$ of DTW. The experimental results verify this.

VI. SUMMARY

In this paper, a time series matching algorithm based on the key points alignment is proposed. It constructs the approximate constraint range based on the alignment relationship of key points, which significantly reduces the computational overhead of distance matrix. The experimental results under multiple time series data sets show that compared with the traditional algorithm, the algorithm in this paper significantly improves the computing speed while maintaining good matching accuracy.

REFERENCES

[1] Pal S H, Patet J N. Time-series data mining: A review[J]. Binary Journal of Data Mining & Networking, 2015, 5(1): 01-04.

[2] Kirlić A, Hadžić M. Big data and time series: A literature review paper[J]. Univerzitetska misao-časopis za nauku, kulturu i umjetnost, Novi Pazar, 2017 (16): 139-146.

[3] Susto G A , Cenedese A , Terzi M . Time-Series Classification Methods: Review and Applications to Power Systems Data[J]. Big Data Application in Power Systems, 2018:179-220.

[4] Wu Y, Shen K, Chen Z, et al. Automatic measurement of fetal cavum septum pellucidum from ultrasound images using deep attention network[C]//2020 IEEE International Conference on image processing (ICIP). IEEE, 2020: 2511-2515.

[5] Gorecki T, Luczak M. Non-isometric transforms in time series classification using DTW[J]. Knowledge-Based Systems, 2014, 61(may): 98-108.

[6] Chen L, Tamer Özsu M, Oria V. Robust and fast similarity search for moving object trajectories [C]// Acm Sigmod International Conference on Management of Data. ACM, 2005, 491-502.

[7] Gorecki, Tomasz. Classification of time series using combination of DTW and LCSS dissimilarity measures[J]. Communications in statistics, B. Simulation and computation, 2018, 47(1-2):263-276.

[8] Hajihashemi Z, Popescu M. A multidimensional time series similarity measure with applications to eldercare monitoring[J]. IEEE Journal of Biomedical & Health Informatics, 2016, 20(3):953-962.

[9] Jiang G, Wang W, Zhang W. A novel distance measure for time series: Maximum shifting correlation distance[J]. Pattern recognition letters, 2019, 117(JAN.):58-65.

[10] Zhang M, Pi D. A New Time Series Representation Model and Corresponding Similarity Measure for Fast and Accurate Similarity Detection[J]. IEEE Access, 2017, 5:1-1.

[11] Chen H, Gao X. Similarity Measurement and Cluster Analysis of Time Series Based on Fluctuation Features[J]. Statistics & Decision, 2019.

[12] Gharghabi S, Imani S, Bagnall A, et al. Matrix profile xii: Mpdist: a novel time series distance measure to allow data mining in more challenging scenarios[C]//2018 IEEE International Conference on Data Mining (ICDM). IEEE, 2018: 965-970.

[13] Kim S W, Park S, Chu W W. An index-based approach for similarity search supporting time warping in large sequence databases[C]// Proceedings 17th international conference on data engineering. IEEE, 2001: 607-614.

[14] Izakian H , Pedrycz W , Jamal I . Fuzzy clustering of time series data using dynamic time warping distance[J]. Engineering Applications of Artificial Intelligence, 2015, 39(mar.):235-244.

[15] Keogh E, Wei L, Xi X, et al. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures[J]. VLDB Journal. 2009, 18: 611-630.

[16] Rakthanmanon T, Campana B, Mueen A, et al. Searching and mining trillions of time series subsequences under time warping[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM, 2012.

[17] Cai Q, Chen L, Sun J. Piecewise statistic approximation based similarity measure for time series[J]. Knowledge-Based Systems, 2015, 85: 181-195.

[18] Cuturi M, Blondel M. Soft-dtw: a differentiable loss function for time-series[C]//International conference on machine learning. PMLR, 2017: 894-903.

[19] Geler Z , Kurbalija V , Ivanovic M , et al. Dynamic Time Warping: Itakura vs Sakoe-Chiba[C]// 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA). IEEE, 2019.

[20] Geler Z, Kurbalija V, Radovanovi M, et al. Impact of the Sakoe-Chiba band on the DTW time series distance measure for kNN classification[C]// International Conference on Knowledge Science, Engineering and Management. Springer, Cham, 2014.

[21] Keogh E , Chakrabarti K , Pazzani M , et al. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases[J]. Knowledge and Information Systems, 2001, 3(3):263-286.

[22] Chen Y, Keogh E, Hu B, Begum N, et al. The UCR Time Series Classification Archive. 2018. www.cs.ucr.edu/~eamonn/time_series_date

Auto-Encoding GAN for Reducing Mode Collapse and Enhancing Feature Representation

Xiaoxiang Lu, Yang Zou*, Xiaoqin Zeng, Xiangchen Wu, Pengfei Qiu
Institute of Intelligence Science and Technology, School of Computer and Information,
Hohai University, Nanjing, China
{luxx0824, yzou, xzeng, wxc}@hhu.edu.cn

***Abstract*—Generative Adversarial Nets (GAN) has been a popular research topic in processing of images, speech, texts, and videos, and many other fields. However, GAN still has some drawbacks such as unstable training and mode collapse. To address these challenges, this paper proposes an auto-encoding GAN, which is composed of a set of generators, a discriminator, an encoder and a decoder. A set of generators is responsible for learning different modes, accelerating the convergence of the model and preventing model collapse. The discriminator is used to distinguish between real samples and generated ones. In order to improve feature representation of the encoder and prevent multiple generators from covering a certain mode, an approach consisting of three phases is proposed accordingly. First, a clustering algorithm is presented to perceive the distribution of real and generated samples. Then, cluster center matching is utilized to keep consistency of the distribution of real and generated samples. Finally, the encoder and decoder are jointly optimized by the generated and real samples. Therefore, the encoder can map the generated and real samples to the embedding space so as to encode distinguishable features, and the decoder can distinguish from which generator the generated samples come and from which mode the real samples come. Experiments are conducted on image datasets to verify effectiveness of the auto-encoding GAN for reducing mode collapse and enhancing feature representation.**

Keywords-GAN; mode collapse; feature representation; cluster

I. INTRODUCTION

GAN [1] is a generative model proposed by Goodfellow, and it has a wide range of applications in data generation, style transfer, image inpainting, etc. [2,3,4,5]. However, GAN suffers from several problems such as unstable training, blurred images and mode collapse, etc. Among them, mode collapse has been a crucial problem that needs to be addressed for GAN. Mode collapse means that GAN only generates some of all modes, and others are missing. In order to reduce mode collapse, researchers have made a number of improvements, which can be summarized in the following four classes.

1) Adding constraints: Unrolled GAN [6] stabilizes the training of GAN by K-step cyclic training to avoid the generator falling into a single mode. CGAN [7] strengthens the relationship between input and output data by using conditional information, forcing the generator to learn different modes. InfoGAN [8] maximizes the mutual information between the input data and the hidden code to obtain interpretable and distinguishable features and avoid mode collapse. Mixture Density GAN [9] encourages the discriminator to conform to Gaussian mixture distribution in the embedded space, which ensures that the generator fits Gaussian mixture distribution as much as possible, covering different modes.

2) Adding generators: The typical models are MADGAN [10] and MGAN [11]. Unlike GAN, their discriminators not only need to distinguish between real samples and generated samples, but also distinguish from which generator the generated samples come.

3) Modifying loss function: WGAN [12] uses the EM distance to measure the difference between generated distribution and real distribution, avoiding the problem of gradient disappearance of the generator, and making sure the generated distribution is as close to real distribution as possible to cover multiple modes.

4) Imposing gradient penalty: DRAGAN [13] imposes a gradient penalty on the training samples to the discriminator, and tries to construct a linear function on the training samples to find the global optimal solution.

In classes *1)* and *2)*, the discriminator only considers the distribution of the generated samples, but neglects the distribution of the real samples, which may lead to a problem that multiple generators cover different parts of a certain mode, resulting in mode collapse. As for *3)* and *4)*, although the generator can cover multiple modes, it still retains the characteristics of continuous mapping in GAN, which may cover the blank area between modes and generate poor samples, resulting in training instability.

In order to address the issues of modes collapse and training instability, this paper proposes an auto-encoding GAN, which consists of a set of generators, a discriminator, an encoder and a decoder. The network architecture is shown in Fig. 1, where a set of generators is responsible for covering different modes, and the discriminator, like GAN, is used to distinguish between real samples and generated samples, ensuring that the generated distribution does not deviate from real distribution. In order to prevent multiple generators from covering different parts of a certain mode, different from the above-mentioned multi-generator model, a clustering algorithm is introduced to perceive

*Corresponding author: yzou@hhu.edu.cn (Y. Zou)

DOI reference number: 10.18293/SEKE2022-152

the distribution of real samples and generated samples, and an algorithm of cluster center matching is presented to keep consistency of the distribution of real and generated samples. Then, the encoder and decoder are jointly optimized by the generated and real samples. Therefore, the encoder can map generated samples and real samples to the embedding space so as to encode distinguishable feature, and the decoder can distinguish from which generator the generated samples come and from which mode the real samples come. Experimental results show that the trained auto-encoding GAN can not only reduce mode collapse, but also have preferable capability in feature representation.

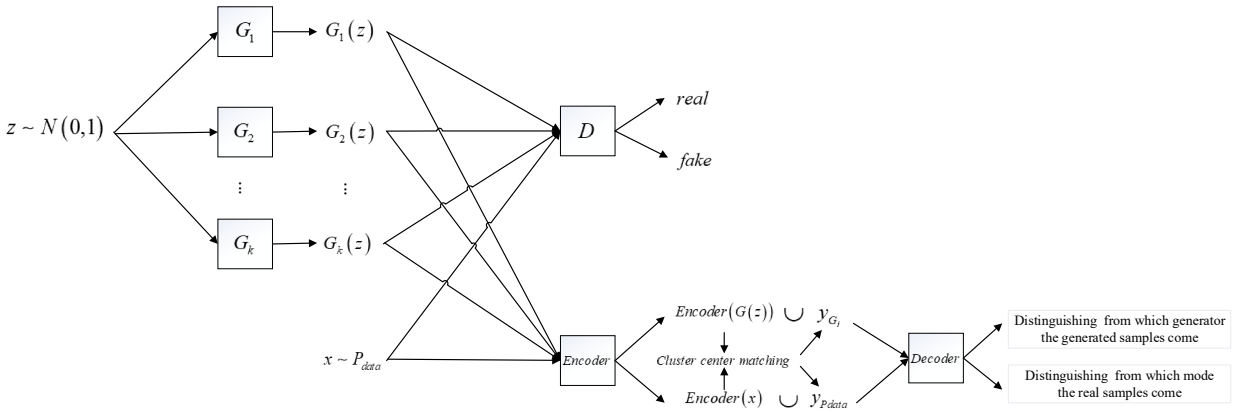


Figure 1. Architecture of Auto-Encoding GAN. $z \sim N(0,1)$ is input to different $G_1, G_2, ..., G_k$ in equal amounts. The generated samples $G_{i\ (1 \le i \le k)}(z)$ and real samples x are input to the discriminator and encoder-decoder respectively, and the discriminator distinguishes the real and the fake and the encoder extract features. The pseudo labels y_{data} and y_{G_i} of real and generated samples are determined by the cluster center matching algorithm. The decoder distinguishes from which generator the generated samples come and from which mode the real samples come by cross-entropy loss.

The technical contributions of the paper are summarized as follows:

1) From the perspective of data distribution, it verifies that adding generators is an effective way to tackle training instability and mode collapse.

2) An algorithm for cluster center matching is proposed to keep consistency of the distribution of real and generated samples, prevent multiple generators from covering different parts of a certain mode and reduce mode collapse.

3) The encoder and decoder are jointly optimized by generated samples and real samples, which reduces the mode collapse of generator and enhancing the feature representation of encoder.

II. AUTO-ENCODING GAN

This section elaborates on the auto-encoding GAN's network architecture, objective function and algorithm of cluster center matching in detail.

A. Adding Generators to Reduce Training Instability and Mode Collapse

The training instability of GAN means that the learning processes of generator G and discriminator D are difficult to converge together. From the perspective of data distribution, the reason is that the complexity of the real samples' distribution affects the speed of G and D convergence. If the real samples are a single-mode simple distribution, G and D converge together soon. If the real samples are a multi-mode complex distribution, G and D are difficult to converge together, as shown in the Fig. 2 and Fig. 3.

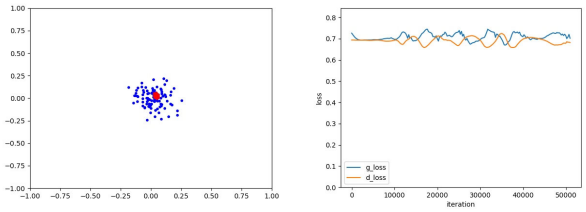


Figure 2. Left is mode coverage of GAN on single-mode simple distribution, where blue and red points represent real and generated samples, respectively. Right is the loss curve of G and D of GAN on the single-mode distribution.

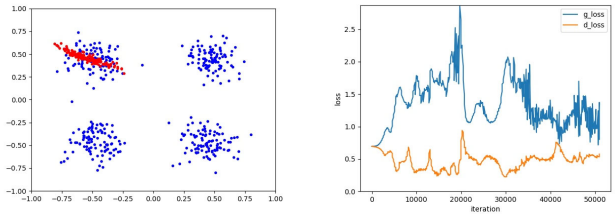


Figure 3. Left is mode coverage of GAN on four-mode complex distribution, where blue and red points represent real and generated samples respectively. Right is the loss curve of G and D of GAN on the four-mode distribution.

As shown in Fig. 3, it can be seen from the loss curve of four-mode complex distribution that the training of G and D is unstable and difficult to converge together. However, in Fig. 2, G and D of GAN converge easily together for single-mode simple distribution. Obviously, if one generator is used to learn multiple modes, it will be difficult to converge and the training is unstable. Therefore, if a set of generators is employed to learn multiple modes, trying to ensure that a single generator covers a single mode, the training instability of GAN can be effectively

settled, as shown in Fig. 4. Accordingly, it can be seen from Fig. 3 and Fig. 4 that the generators and discriminator of auto-encoding GAN converge and tend to be stable much more quickly than that of GAN.

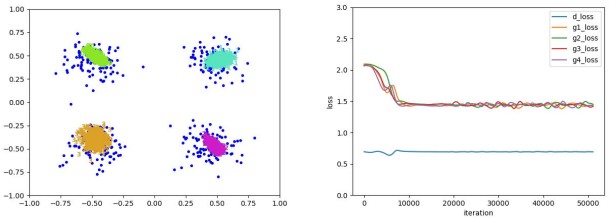


Figure 4. Left is mode coverage of Auto-Encoding GAN on four-mode complex distribution, where blue points represent real samples, and others represent generated samples in which samples generated from different generators are denoted by different colors. Right is the loss curve of G and D of Auto-Encoding GAN on the four-mode distribution.

The mode collapse of GAN refers to that GAN tends to concentrate continuously mapped values on a single mode during the training process. The main reason for the phenomenon is GAN can only approximate continuous mapping, on the contrary, but the multi-mode distribution belongs to the discrete distribution that is not continuous, so GAN is difficult to cover discrete multi-mode distributions with continuous map. If a continuous map is forced to cover all modes, the values of the continuous map will inevitably cover some blank areas outside the mode, so GAN may generate some samples that have no realistic meaning, which explains why GAN may generate some poor samples. For a theoretical proof of GAN's mode collapse, please refer to [14].

Therefore, the key to addressing mode collapse is to build a discontinuous map. Adding generators can discretize the generation distribution, so that each generator covers a mode, which is essentially similar the way that multiple GAN utilizes to achieve the discontinuous mapping of multi-mode distribution, but the difference is that simply adding generators can do.

In summary, auto-encoding GAN adopts the way of adding generators to realize discontinuous mapping of multi-mode distribution, which can not only speed up convergence of the model, but also cover multiple modes.

B. *Objective Function of Auto-Encoding GAN*

Assume real samples is subject to distribution of real samples $x \sim P_{data}$, where P_{data} is the distribution of real samples, containing k modes, and noise samples is subject to standard normal distribution $z \sim N(0,1)$. According to the architecture of the auto-encoding GAN, k generators $G = \{G_1, G_2, ..., G_k\}$ are designed to try to cover various k modes. A discriminator D is responsible for distinguishing between real and generated samples. An *Encoder* is used to map the generated samples and real samples to the embedding space. A *Decoder* not only distinguishes from which generator the generated samples come, but also distinguishes from which mode the real samples come, where G , D , *Encoder-Decoder* (ED) are deep networks. y_{G_i} represents the label corresponding to the generated samples by the *i-th* generator, and $y_{P_{data}}$ represents the label corresponding to the real sample, which is mainly a pseudo-label after cluster center matching. The purpose of auto-encoding GAN training is to obtain a set of $G = \{G_1, G_2, ..., G_k\}$ that can cover different modes and an *Encoder* with capability of feature representation. Therefore, the objective function that needs to be optimized for auto-encoding GAN can be written as follows.

$$\min_{G,ED}\max_{D} L(G,D,ED) = E_{x\sim P_{data}}\left[log\left(D(x)\right)\right] + \sum_{i=1}^{k} E_{z\sim P_{G_i}}\left[log\left(1-D\left(G_i(z)\right)\right)\right] - \lambda\left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}\left[log\left(ED\left(G(z)\right)\right)\right] + y_{P_{data}} E_{x\sim P_{data}}\left[log\left(ED(x)\right)\right]\right] \quad (1)$$

where λ is the weight to keep the balance between D and ED .

Assume G , D , and ED have sufficient capacity and training time, the conditions of convergence of them are given below.

Proposition 1. For G fixed, the optimal D is

$$D(x) = \frac{P_{data}}{P_{data} + \sum_{i=1}^{k} P_{G_i}}$$

Proposition 2. For G fixed, the optimal ED is

$$\begin{cases} ED_G\left(G(z)\right) = \sum_{i=1}^{k} y_{G_i} \\ ED_x(x) = y_{p_{data}} \end{cases}$$

For the convenience of description, ED can be divided into $ED_G(x)$ and $ED_x(x)$, which are responsible for the representation and classification of generated samples and real samples, respectively.

Obviously, when $P_{data} = \sum_{i=1}^{k} P_{G_i}$, D converges to the optimum. At this time, it can be deduced that $G(z) = x$, that is $\sum_{i=1}^{k} y_{G_i} = y_{p_{data}}$. When $\sum_{i=1}^{k} y_{G_i} = y_{p_{data}}$, it can be deduced that $ED_G\left(G(z)\right) = ED(x) = y_{p_{data}}$, so the convergence condition of ED is also reached.

Proposition 3. For the optimized D and ED fixed, the optimized generator G is

$$G = -log4 + 2JSD\left(P_{data} \,||\, \sum_{i=1}^{k} P_{G_i}\right) - 2\lambda \int_x \sum_{i=1}^{k} y_{G_i} log\left(y_{G_i}\right)$$

When $P_{data} = \sum_{i=1}^{k} P_{G_i}$, $JSD\left(P_{data} \,||\, \sum_{i=1}^{k} P_{G_i}\right) = 0$ .since

both y_{G_i} and y_{Pada} are known, the generator G converges to

$$G = -log4 - 2\lambda \int_x \sum_{i=1}^{k} y_{G_i} log\left(y_{G_i}\right) .$$

Due to limited space, the proofs of convergence of D, ED and G are omitted, which is similar to that of GAN.

C. Minimizing Cluster Center Matching

If the decoder only distinguishes from which generator the generated samples come, it may cause the generated samples to cover different parts of a certain mode, resulting in modes collapse. Here, the training results of the InfoGAN and MGAN on 2D dataset are taken as an example to show the case where the modes collapse, as shown in the Fig. 5.

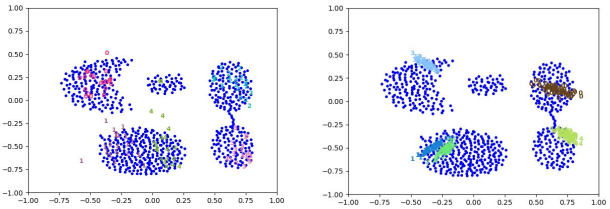


Figure 5. Left is the results of InfoGAN on 2D dataset, and right is the results of MGAN on 2D dataset.

If the mode distribution of real samples is taken into account, the classifier can effectively avoid mode overlap. However, the mode distribution of real samples is unknown, so how to approximate the distribution of real samples is a key issue. Both the generated distribution and real distribution are distributions of the embedding space output by the encoder. Since the same mode is usually clustered in the embedding space due to the similarity between real samples, a clustering algorithm is employed to perceive the mode distribution of real samples.

At the same time, in order to prevent perceived results from deviating from the real distribution, we take the generated distribution as a reference, and balance the deviation by minimizing the cluster centers matching of the generated distribution and real distribution. The reason why the generated distribution can be used as a reference is that it is constantly approaching the real distribution during the training process. Theoretically, when the model converges, the generated distribution approximates the real distribution. However, in actual training the encoder cannot accurately learn the characteristics of the real distribution, since the generated distribution cannot fully represent the real distribution. If some prediction information of the real distribution is added, the learning capability of the encoder for the real distribution will be considerably improved.

According to the above analysis, assuming that the generated samples and the real samples are fed into the encoder, the embedded features output by the encoder are respectively $h_x = Encoder(x)$ and $h_{G(z)} = Encoder(G(z))$. The cluster centers of real samples and generated samples obtained by the clustering algorithm are $\mu_x = \{\mu_{x,1}, \mu_{x,2}, ..., \mu_{x,k}\}$ and $\mu_{G(z)} = \{\mu_{G_1(z),1}, \mu_{G_2(z),2}, ..., \mu_{G_k(z),k}\}$. In order to minimize the matching of the cluster centers of the real distribution and the generated distribution, the loss function that needs to be satisfied is given as follows.

$$minL_c(G, Encoder) = \sum_{i=1}^{k} E_{x\sim P_{data}} \left\| h_x - \mu_{x,i} \right\|^2 + E_{z\sim P_{G_i}} \left\| h_{G_i(z)} - \mu_{G_i(z),i} \right\|^2 + \left\| \mu_{x,i} - \mu_{G_i(z),i} \right\|^2 \quad (2)$$

Combined with the loss function, the process of minimizing cluster center matching is introduced in detail, which includes the following three steps.

1) First, the encoder is utilized to take the generated and real samples as input, and outputs the embedded features. Then, k-means++ algorithm is used to cluster the generated samples and real samples in the embedded space respectively, and obtain the clustering center sets μ_x and $\mu_{G(z)}$ of the generated samples and real samples respectively.

2) To minimize the cluster center matching, the distance matrix between the center sets μ_x and $\mu_{G(z)}$ is calculated, and *i-th* $\mu_{x,i}$ the closest matching center $\mu_{G(z),i}$ is found to ensure $\left\| \mu_{x,i} - \mu_{G_i(z),i} \right\|^2$ is the smallest.

3) To unify the cluster assignment of the real samples and the generated samples, the generated samples are used as a reference to keep the matching center pairs $\left(\mu_{x,i}, \mu_{G_i(z),i}\right)$ consistent with the corresponding cluster labels.

After the above process, the current loss of centers matching between real distribution and the generated distribution can be obtained, and the generator and encoder can be further optimized.

D. Training of Auto-Encoding GAN

Combined with the above introduction of auto-encoding GAN, the specific algorithm of auto-encoding GAN is given.

Algorithm 1 Training of Auto-Encoding GAN

Input: Dataset: X, Number of generator: k, Training epochs: n_epochs,

Output: Generator: G, Encoder-Decoder: ED

Initialize generator G, discriminator D, Encoder-Decoder: ED.
for $epoch = 1$ to n_epochs **do**
  Samples a batch x_i from X.
  Samples $z \sim N(0,1)$.
  Calculate $L_D = E_{x\sim P_{data}}[log(1 - D(x))] + \sum_{i=1}^{k} E_{z\sim P_{G_i}}[log(1 - D(G_i(z)))]$ according to $Eq.(1)$.
  Update D by descending along their gradient $\nabla_D(L_D)$.
  Clustering $h_x, h_{(G(z))}$ using k-means++ to get clustering centers $\mu_x, \mu_{G(z)}$.
  Matching μ_x the nearest cluster centers from $\mu_{G(z)}$.
  Setting clustering label $y_G, y_{p_{data}}$ of $h_x, h_{(G(z))}$ according to matching result.
  Calculate $L_{ED} = \left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}[log(ED(G_i(z)))] + y_{P_{data}} E_{x\sim P_{data}}[log(ED(x))]\right]$.
  Update ED by descending along their gradient $\nabla_{ED}(L_{ED})$.
  for $i = 1$ to k **do**
    Calculate $L_{G_i} = \sum_{i=1}^{k} E_{z\sim P_{G_i}}[log(1 - D(G_i(z)))] - \lambda\left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}[log(ED(G_i(z)))]\right]$.
    Updata G_i by descending along their gradient $\nabla_{G_i}(L_{G_i})$.
  end for
end for

III. EXPERIMENTS

In this section, we conduct experiment on image datasets and demonstrate the effectiveness and of auto-encoding GAN for reducing mode collapse and enhancing feature representation.

A. Implementation Details

For the convenience of experiment reproduction, we provide the experimental details. In the process of cluster center matching, it is necessary to obtain the cluster center for matching by k-means++ algorithm (or GMM). The generated and real samples are clustered in each iteration, where k-means++ adopts default parameters of a python package "sklearn" except for the number of clusters that need to be specified.

It can be seen from the loss function that the hyperparameter involved in this paper is mainly λ, and it is used to balance the weight between the discriminator and the encoder-decoder. Their losses can be unified to magnitude according to the actual training value. In our experiments, λ set to 0.5. In addition, k determines the number of generators, and is generally equal to the number of categories of the dataset in experiments.

B. Experiments of Image Datasets

In this subsection, we conduct experiments on the image datasets to verify the effectiveness of auto-encoding GAN. We choose some frequently used datasets, USPS, MNIST, Fashion-MNIST, Coil-20, and Cifar-10, to conduct experiments.

Image Datasets. USPS and MNIST [15] are digital image datasets. USPS contains 9298 images (16×16) of 10 categories, where 7291 images for training and 2007 images for testing. MNIST contains 70000 images (28×28), including 60000 training images and 10000 testing images, with 10 categories. Fashion-MNIST is similar to MNIST, except that it contains different categories. Coil-20[16] contains 1440 images (128×128), with 20 categories. Cifar-10[17] involves 60,000 color images (32×32) of 10 categories, including 50000 for training and 10000 for testing. Only the training images is used in the experiments.

Evaluation Indicators. The generator is evaluated from the quality and diversity of the generated images, and FID [18] is often used to serve this purpose. The lower the FID score, the closer the generated distribution is to the real distribution, which means that the quality of the generated images is higher and the diversity is better.

As for the evaluation of the encoder-decoder, the feature representation of the encoder is verified from the classification of images. Clustering accuracy (ACC) and normalized mutual information (NMI) is usually exploited as the evaluation indicators of feature representation (especially unsupervised). The larger the value of ACC and NMI, the better the capability of feature representation.

Network Architecture. For USPS, MNIST and Fashion-MNIST datasets, both the generator and discriminator choose the DCGAN network architecture. The encoder has the same convolution architecture as the discriminator. The decoder is coded as a fully connected layer, and the activation function is softmax for classification of generated and real samples. The other parameters of the above datasets are set as follows: the optimizer is uniformly Adam, the learning rate is 0.0004, the batch size is 128, and the epochs is 500.

Experimental Results. Comparative experiments are implemented from the coverage degree of the mode and the feature representation of the model on image datasets, and each class is regarded as a mode in the image datasets. The models selected for comparison are WGAN, DCGAN, InfoGAN and MGAN, as they are representative in each category and more relevant to our model.

1) Experiments are implemented for the mode coverage of different models on the image datasets. The degree of model coverage is mainly evaluated by the FID. The experimental results are shown in Table I.

TABLE I. FID (lower is better) of different models on Image Datasets

Datasets / Models	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
DCGAN	51.41	28.70	72.78	41.39	95.47
WGAN	60.74	83.86	82.79	57.46	100.25
InfoGAN	34.06	26.71	72.92	37.41	**80.82**
MGAN	30.85	25.29	81.24	36.92	87.23
Ours	**28.42**	**22.07**	**64.63**	**35.27**	85.46

From the experimental results in Table I, it can be concluded that auto-encoding GAN is better than other models on USPS, MNIST, Fashion-MNIST and Coil-20. On Cifar-10, InfoGAN is the best and better than auto-encoding GAN.

From the average of FID over all datasets, auto-encoding GAN is 10.77 lower than DCGAN, 27.06 lower than WGAN, 4.01 lower than InfoGAN, and 4.53 lower than MGAN. It is further demonstrated that auto-encoding GAN is significantly better other models in reducing mode collapse.

In addition to the above FID comparison experiments, some images generated by various models on MNIST are shown in Fig. 6.

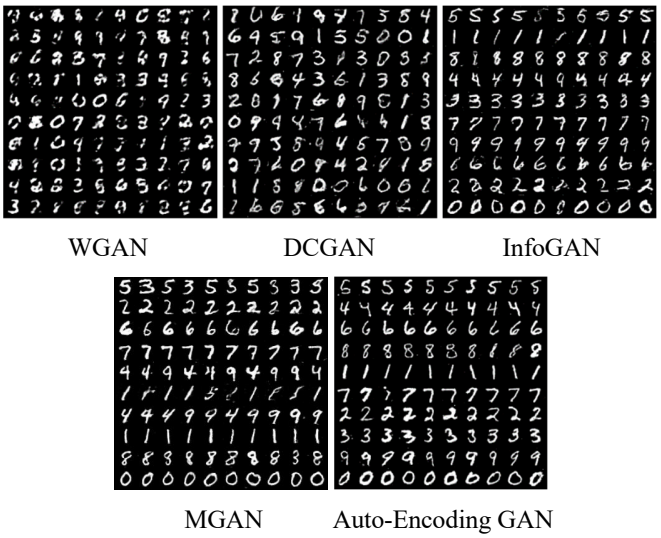


Figure 6. Images generated by various models on MNIST

From the generated images on MNIST, it can be seen that the quality of the images generated by WGAN and DCGAN is poor, and the images are generated in a random way. Both

InfoGAN and MGAN generate a variety of modes, but partial modes have a phenomenon of overlap, such as '1' of MGAN, and the purity of the generated images is low, such as '9' and '4'. Compared with InfoGAN and MGAN, auto-encoding GAN generate all modes, and the purity of generated images is higher.

2) Experiments are implemented for feature representation capabilities of different models on the image datasets. Because WGAN and DCGAN do not have multi-mode representation abilities, only InfoGAN, MGAN and auto-encoding GAN are selected in the experiments. NMI and ACC are used to evaluate the feature representation capabilities of a model. The experimental results are shown in Table II and Table III.

TABLE II. NMI (%) (higher is better) of different models on Image Datasets

Datasets / Model	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
InfoGAN	72.57	85.27	59.49	74.70	**20.49**
MGAN	75.99	85.01	54.11	66.49	17.26
Ours	**76.21**	**87.24**	**59.83**	**79.34**	18.34

TABLE III. ACC (%) (higher is better) of different models on Image Datasets

Datasets / Model	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
InfoGAN	71.42	92.64	**58.44**	60.41	**34.80**
MGAN	74.13	92.52	56.09	56.53	32.17
Ours	**74.92**	**93.27**	58.11	**68.94**	33.34

From the experimental results in Table II, it can be concluded that auto-encoding GAN is better than other models on USPS, MNIST and Coil-20 and Fashion MNIST. On Cifar-10, InfoGAN performs better than auto-encoding GAN. The possible reason for this is InfoGAN can randomly select the hidden code to prevent overfitting of the model compared with multi-generator models.

From the average of NMI over all datasets, auto-encoding GAN is 1.69% better than InfoGAN, and 4.22% better than MGAN. It is further demonstrated that auto-encoding GAN have preferable capability of feature representation.

IV. Conclution

This paper proposed an auto-encoding GAN to reduce the mode collapse of the generator and enhance feature representation of the encoder. It consists of a set of generators, a discriminator, an encoder and a decoder. A set of generators is responsible for learning different modes, accelerating the convergence of the model and preventing mode collapse. The discriminator is used to distinguish between real samples and generated samples. The encoder maps the generated samples and real samples to the embedding space, encoding distinguishable feature information among modes. The decoder distinguishes from which generator the generated samples come and from which mode the real samples come. Different from other multi-generator models, in order to improve the feature representation of the encoder and prevent multiple generators from covering a certain mode, an approach consisting of three phases is proposed accordingly. First, a clustering algorithm is presented to perceive the distribution of real and generated samples. Then, cluster center matching is utilized to keep consistency of the distribution of real and generated samples. Finally, the encoder and decoder are jointly optimized by the generated and real samples. We have conducted experiments on image datasets to fully demonstrate the effectiveness of auto-encoding GAN in reducing mode collapse and enhancing feature representation.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley and S. Ozair. “Generative adversarial nets,” Advances in neural information processing systems, 2014.

[2] A. Radforda, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[3] T. Karras, S. Laine and T. Aila. “A style-based generator architecture for generative adversarial networks,” In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401-4410, 2019.

[4] JY. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” In Proceedings of the IEEE international conference on computer vision, pp. 2223-2232, 2017.

[5] D. Pathak, P. Krahenbuhl, J. Donahue J, T. Darrell and A.A. Efros, “Context encoders: Feature learning by inpainting”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2536-2544, 2016.

[6] L. Metz, B. Poole, D. Pfau , and J. Sohl-Dickstein, “Unrolled generative adversarial networks,” arXiv preprint arXiv:1611.02163, 2016.

[7] M Mirza and S. Osindero, “Conditional generative adversarial nets,” Computer Science, arXiv preprint arXiv:1411.1784, 2014.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” Advances in neural information processing systems, 2016.

[9] H. Eghbal-Zadeh, W. Zellinger, and G. Widmer, “Mixture density generative adversarial networks,” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5820-5829, 2019.

[10] A. Ghosh, V. Kulharia, V. Namboodiri, V. P. Namboodiri, P. H. Torr, and P. K. Dokania, “Multi-agent diverse generative adversarial networks,” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8513-8521, 2018.

[11] Q. Hoang, T.D. Nguyen, T. Le, and D. Phung, “MGAN: Training generative adversarial nets with multiple generators,” In International conference on learning representations, February 2018.

[12] M. Arjovsky, S. Chintala, and L.Bottou, “Wasserstein generative adversarial networks,” In International conference on machine learning, pp. 214-223, July 2017.

[13] L. Tran, X. Yin, and X. Liu, “Disentangled representation learning gan for pose-invariant face recognition,” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1415-1424, 2017.

[14] Y. Guo, D. An, X. Qi, Z. Luo, S. T. Yau, and X.Gu, “Mode collapse and regularity of optimal transportation maps,” arXiv preprint arXiv:1902.02934., 2019.

[15] Y. Lecun, and L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,”. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[16] S. Nene, S. Nayar, and Murase, “Columbia object image library (coil-20),” Technical Report, 1996.

[17] A. Krizhevsky, and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” Advances in neural information processing systems, pp. 6629-6640, 2017.

An Explainable Knowledge-based AI Framework for Mobility as a Service

Enayat Rajabi[1, 2], Sławomir Nowaczyk[1], Sepideh Pashami[1], and Magnus Bergquist[1]

[1]Center for Applied Intelligent Systems Research, Halmstad University, Halmstad, Sweden
{*enayat.rajabi,slawomir.nowaczyk,sepideh.pashami, magnus.bergquist*}*@hh.se*
[2]Shannon School of Business, Cape Breton University, Canada
enayat_rajabi@cbu.ca

Abstract

Mobility as a Service (MaaS) is a relatively new domain where new types of knowledge systems have recently emerged. It combines various modes of transportation and different kinds of data to present personalized services to travellers based on transport needs. A knowledge-based framework based on Artificial Intelligence (AI) is proposed in this paper to integrate, analyze, and process different types of mobility data. The framework includes a knowledge acquisition process to extract and structure data from various sources, including mobility experts and add new information to a knowledge base. The role of AI in this framework is to aid in automatically discovering knowledge from various data sets and recommend efficient and personalized mobility services with explanations. A scenario is also presented to demonstrate the interaction of the proposed framework's modules.

1 Introduction

In recent years, with an increasing number of transport services offered in cities and the advancements in technology, an innovative Mobility as a Service (MaaS) concept was introduced [7]. MaaS combines different transportation modes to deliver users' various services based on transport needs, including trip planning, reservation, and payments, through a single interface [8]. The main goal of MaaS is to make commuting convenient for travellers and offer them flexible, priceworthy, reliable, and sustainable mobility services for goods shipping and delivery. Integrating various services and systems such as electronic ticketing, booking, route planning, and payment services across different modes of transportation can make this happen, as part of Smart Cities transformation [13].

Artificial Intelligence (AI) is also increasingly used these days in MaaS to develop advanced mobility services [5] leveraging both spatial (location-based) and temporal detail recorded frequently by devices such as smartphones, micro-mobility vehicles, on-board vehicle computers, or app-based navigation systems to improve traffic flow or transportation logistics. One of the prerequisites of using AI models in a multi-stakeholder domain such as transportation is to provide explainability and the possibility of tracing back the decisions made to their sources. It is crucial for building trust and adoption of AI systems in settings where transparency is required due to high-stakes scenarios [6].

The interpretation of huge amount of data collected from several sources can not be achieved without the presence of a knowledge-based AI system. For example, if a knowledge-based AI system can capture travellers' preferences, it can exclude travel plans or routes of no interest to those users. However, how to obtain this knowledge is one of the key open challenges. Broadly speaking, it can be acquired by extracting and structuring data or information from various sources, including human experts, and storing the data into a knowledge base. The system's knowledge acquisition process may consist of collecting facts, designing rules, concepts, procedures, heuristics formulas, relationships, ontologies, statistics, or other helpful information. Acquiring specific knowledge about travellers allows MaaS to recommend a ranked list of MaaS planes/routes to select the ones that better fit the user's preferences by inferring the similarity of available plans to the user's

DOI reference number: 10.18293/SEKE2022-0020

profile. Knowledge-based AI systems also enable the possibility of providing the right information to the right user with understandable explanations. In this study, we investigate how one can combine different information sources with an understanding of the traveller's context to present a personalized service to travellers based on the users' preferences. The article aims at proposing a knowledge-based AI framework to provide personalized and explainable mobility services to travellers, service providers, drivers, and other mobility users. This framework covers procedures for data collection, knowledge extraction, inference, recommendations and explanations.

2 Related Works

Understanding what makes MaaS particularly challenging is the first step toward identifying the essential features required of the proposed framework in this study. AI has been generally used successfully in many different settings in the field of transportation over the years; see, for example [14, 10, 2, 15].

A recent study [9] proposed an AI model to predict the traffic intensity before the vehicles reach the intersection. The vehicle trajectory data was collected from GPS sensors, longitudinal, lateral and yaw motion, heading and speed of automobile movements. The vehicles with similar conditions were clustered to provide a route planner to users. In terms of traffic flow prediction, Li and Xu [11] developed a short-term traffic flow prediction model based on Support Vector Regression (SVR) to improve the accuracy of traffic flow prediction systems on the California Highway Performance Evaluation System (PeMS) videos. AI-based models are also used to classify driving styles, ranging from aggressive to calm. The classification algorithms are widely used to customize driver assistance systems, assess mobility, crash risk, fuel consumption, or emissions, among others. For example, Mohammadnazar et. al [12] extracted volatility measures based on speed, lateral longitudinal acceleration, and temporal driving volatility (using a 3-second time-frame window) from a set of data and used them for cluster drivers (in aggressive, normal, and calm) using K-means and K-medoids methods. Although several studies applied various AI-based models in MaaS, relatively few of them considered leveraging knowledge-based systems in the models to provide personalized and explainable mobility services. Arnaoutaki et al. [1], as an example, proposed a hybrid knowledge-based system that uses constraint programming mechanisms to provide mobility plans to travellers based on their preferences and exclude the routes that do not match those preferences. Close to this study and in conjunction with the other AI-based models, we propose a knowledge-based AI mobility framework that utilizes context information and knowledge of mobility (acquired from travellers and vehicles) to provide personalized mobility services while being interpretable and explainable for both travellers and domain experts.

3 Proposed Framework for Mobility as a Service

There are different sources of mobility knowledge, including contextual data (weather, traffic, disruptions), operational (routes, schedules, business rules, and deliveries), personal (passengers, travellers, and drivers) and transactional (booking, tickets, and payments). We propose a knowledge-based framework (Figure 1) that provides customized, explainable, and enriched services based on various types of mobility data. The framework intends to integrate different mobility data types processes, analyze them, and recommend a real-time personalized service with customized explanations based on mobility users' needs. We will discuss the main modules of this framework briefly below.

3.1 Semantic enrichment

A semantic enrichment module matches concepts in a system with the most appropriate semantic entities available from various sources. It leverages mobility ontologies, vocabularies, and API services to fetch data and integrate similar entities using semantic similarities metrics from different resources. The outcome of this module is an enriched set of entities that are injected into the knowledge base.

3.2 Mobility ontology

Ontology can be seen as a formal representation of entities in a domain, their relations, properties or attributes, and constraints. Different mobility concepts can be defined as terminologies and vocabularies to describe real-world features or phenomena associated with a specific discipline, domain or application and their relationships. Developing mobility ontologies in a knowledge-based AI system facilitates knowledge acquisition and allows knowledge reasoning. The concepts of ambiguity and semantic heterogeneity in mobility systems can be resolved using ontologies in knowledge bases. Leveraging ontologies also solves semantic heterogeneity problems and enriches

DOI reference number: 10.18293/SEKE2022-0020

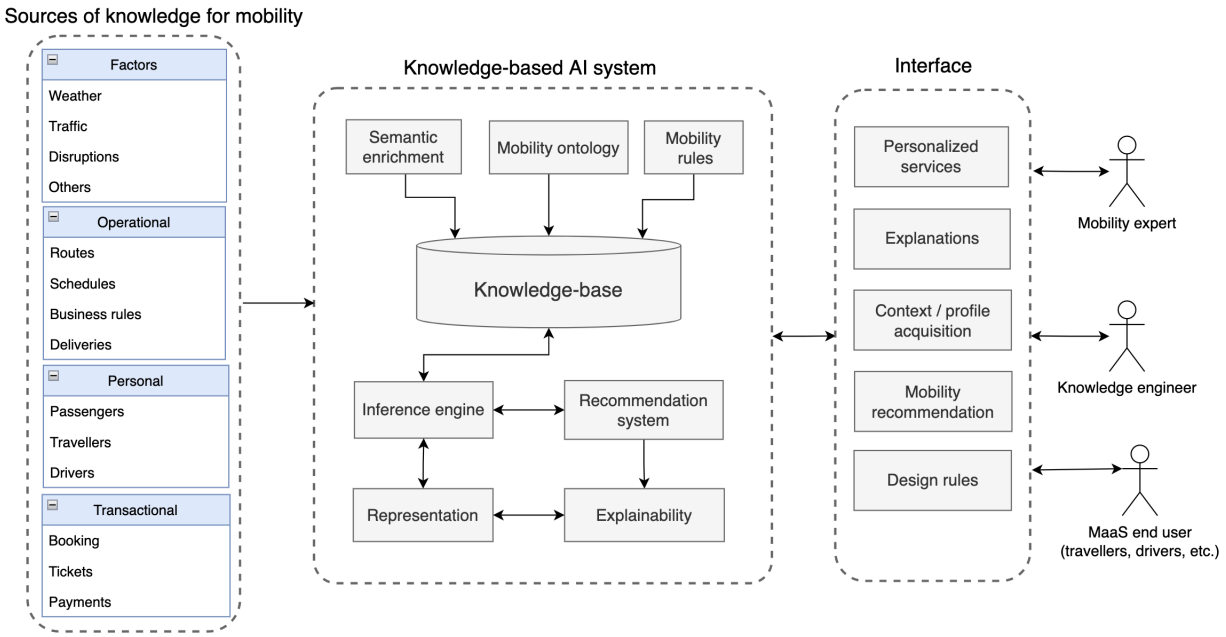


Figure 1. Knowledge-based AI framework for mobility as a service

services' descriptions to make their semantics machine-interpretable and provides efficient search results.

3.3 Rule engine

Ontologies assist in defining new entities, concepts and their relationships. However, information about the context and procedural knowledge are usually represented by logic rules in the form of condition–action pairs: *IF condition holds, THEN perform action.* Conditions usually contain patterns and variables that may be linked to facts. Each rule has a first, matching part, and a second, action one, which modifies the working memory or outputs something. A rule may have variables that are linked to values in the working memory using pattern matching [3]. For example, consider a rule stating that a traveller would not use an e-bike in rainy weather; the variables ?t (traveler), ?e (e-bike), and ?u (uses) are matched to all the available data that satisfies the condition.

3.4 Recommendation system

The recommender systems understand the preferences and needs of MaaS users, their context and their environment to assist them with a personalized mobility service. The recommendation system module of the proposed framework is connected to the inference engine, explainability, and representation modules to suggest a customized and personalized mobility service based on inductive reasoning.

3.5 Representation

The representation module is responsible for preparing the outputs of the recommender system and explainability modules and visualizing them understandably and convincingly for the users. Since different explanations and visualizations might be needed for various users, this module assigns proper explanations and delivers them to the interface layer to be visualized.

3.6 Explainability

Explainability makes an AI system more understandable, transparent and responsible while reducing risks. This module is responsible for justifying the personalized recommendation made by the recommendation system – to both travellers and domain experts. For example, the module explains the reasons for recommending a specific service (e.g., a taxi with an electric car) for a traveller or, in extreme cases, why no recommendations are available for a particular user. One approach toward explainability is using features suggested by experts to bridge the gap between knowledge-driven and data-driven approaches.

DOI reference number: 10.18293/SEKE2022-0020

4 Scenario

The following scenario shows how a knowledge-based AI system can provide a customized and personalized service to two different travellers. As Figure 2 illustrates, the knowledge-based AI system (forming the core of the proposed framework) is connected to contextual and non-contextual sources, such as public transport, taxi, rental car services, and weather data – to capture various information. It also provides a mobile application or wearable device for travellers to address their needs and schedules. With the help of mobility experts and knowledge engineers, a set of rules is created based on the different information and integrated into the knowledge base. These rules are updated regularly based on changing circumstances, trends in mobility patterns, etc.

In this scenario, two travellers, namely Alex and Mary, usually use e-bikes on Wednesdays. However, the system notices that next Wednesday will be rainy, based on weather forecast data. The system uses a rule engine and concludes that travellers cannot ride e-bikes in rainy weather, recommending an alternative transportation solution with an explanation to each traveller. According to the knowledge base, Mary is interested in sustainable mode of transportation and prefers using electric cars, while Alex likes gas automobiles. The system searches the taxi drivers in Mary's area and arranges a taxi service with an electric car for her. It also suggests a gas car for Alex according to his location. Both travellers are connected to the corresponding taxi drivers. If there are no taxi drivers available for these travellers, the AI system might not provide any recommendations and notifies the domain expert or knowledge engineer to add a new rule to the system to expand its functionality.

5 Discussion

Knowledge-based AI systems can make valuable contributions to generating flexible and intelligent solutions; however, there are several challenges due to the complexity of mobility services. Collecting the requirements of MaaS users and accessing real-time data (contextual and non-contextual) from several sources is challenging. Also, updating the proposed knowledge base with, e.g., contextual data such as weather or traffic information requires real-time services to respond to the travellers' up-to-date requirements and needs. Furthermore, connecting different types of data with various structures and identifying their semantic relationships adds another challenge while providing richer explainable services. In terms of semantic enrichment, a semi-automatic approach should be followed to enrich different types of data coming into the system and facilitate interoperability issues in MaaS. A mobility expert with extensive knowledge of mobility data should construct a mobility ontology or adjust existing ontologies in the system and define rules in the system. To answer questions like "what should happen if someone uses e-bikes in rainy weather?", the mobility expert should inject a rule in the knowledge base. The system should provide tools and interfaces to update and optimize the system's rules efficiently.

In terms of explainability, one type of explanation can not be sufficient for different mobility users such as domain experts, knowledge engineers, and Maas end-users. Although recently emerged explainability approaches [4] can address the knowledge engineers' needs, they have not been adapted to handle the requirements of mobility stakeholders and end-users. Recommendations augmented with reasoning and explanations can increase awareness of the framework's performance.

6 Conclusion

In the light of technological advancement in the mobility domain due to widespread AI adoption, individuals demand more personalized transportation solutions. Off the shelf, AI solutions provide pieces of the puzzle to solve transportation needs in MaaS. This paper proposed a knowledge-based AI framework considering all the necessary modules to enable new services in MaaS, including three important modules (knowledge base, recommendation system, and explainability) to provide personalized and explainable services to MaaS users. Although a scenario was presented to support the proposed framework, further investigation is needed concerning the development of its modules and their technical interactions.

References

[1] K. Arnaoutaki, B. Magoutas, E. Bothos, and G. Mentzas. A hybrid knowledge-based recommender for mobility-as-a-service. In *ICETE (1)*, pages 101–109, 2019.

[2] L. Barua, B. Zou, and Y. Zhou. Machine learning for international freight transportation management: a comprehensive review. *Research in Transportation Business & Management*, 34:100453, 2020.

[3] C. Berdier. An ontology for urban mobility. In *Ontologies in Urban Development Projects*, pages 189–196. Springer, 2011.

DOI reference number: 10.18293/SEKE2022-0020

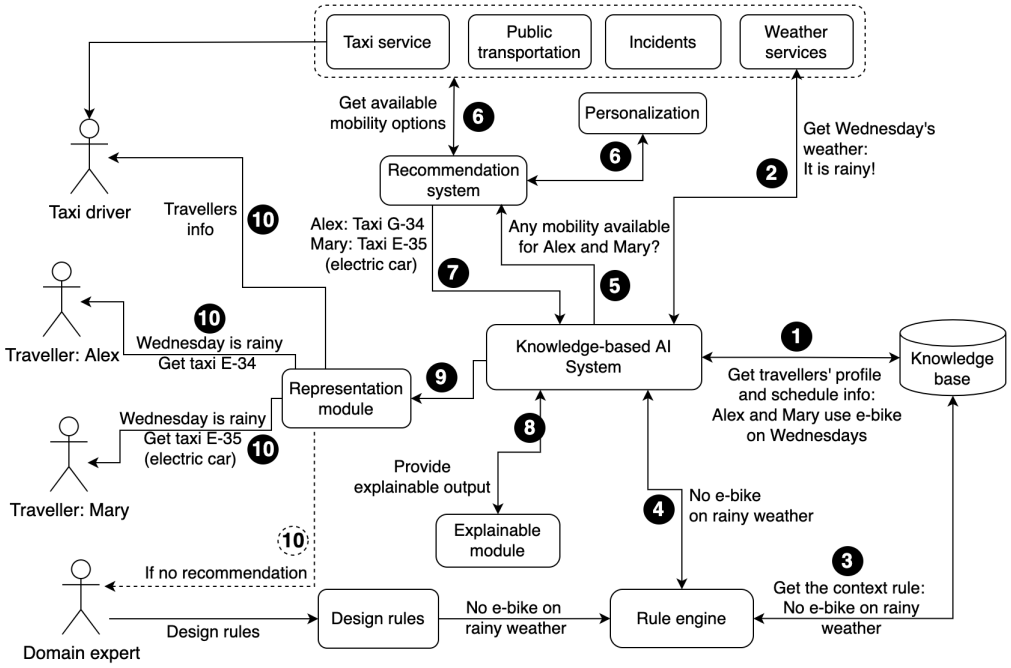


Figure 2. A scenario based on the proposed knowledge-based AI framework

[4] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery.

[5] M.-R. Bouguelia, A. Karlsson, S. Pashami, S. Nowaczyk, and A. Holst. Mode tracking using multiple data streams. *Information Fusion*, 43:33–46, 2018.

[6] K. Gade, S. C. Geyik, K. Kenthapadi, V. Mithal, and A. Taly. Explainable AI in industry. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 3203–3204. Association for Computing Machinery.

[7] W. Goodall, T. Dovey, J. Bornstein, and B. Bonthron. The rise of mobility as a service. *Deloitte Rev*, 20:112–129, 2017.

[8] P. Jittrapirom, V. Caiati, A.-M. Feneri, S. Ebrahimigharehbaghi, M. J. Alonso González, and J. Narayan. Mobility as a service: A critical review of definitions, assessments of schemes, and key challenges. 2017.

[9] S. J. Kamble and M. R. Kounte. On road intelligent vehicle path predication and clustering using machine learning approach. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 501–505. IEEE, 2019.

[10] T. Kim, S. Sharda, X. Zhou, and R. M. Pendyala. A stepwise interpretable machine learning framework using linear regression (lr) and long short-term memory (lstm): City-wide demand-side prediction of yellow taxi and for-hire vehicle (fhv) service. *Transportation Research Part C: Emerging Technologies*, 120:102786, 2020.

[11] C. Li and P. Xu. Application on traffic flow prediction of machine learning in intelligent transportation. *Neural Computing and Applications*, 33(2):613–624, 2021.

[12] A. Mohammadnazar, R. Arvin, and A. J. Khattak. Classifying travelers' driving style using basic safety messages generated by connected vehicles: Application of unsupervised machine learning. *Transportation research part C: emerging technologies*, 122:102917, 2021.

[13] S. Nowaczyk, T. Rögnvaldsson, Y. Fan, and E. Calikus. *Towards Autonomous Knowledge Creation from Big Data in Smart Cities*, pages 1–35. Springer International Publishing, Cham, 2020.

[14] T. Rögnvaldsson, S. Nowaczyk, S. Byttner, R. Prytz, and M. Svensson. Self-monitoring for maintenance of vehicle fleets. *Data Mining and Knowledge Discovery*, 32:344–384, 2018.

[15] N. Servos, X. Liu, M. Teucke, and M. Freitag. Travel time prediction in a multimodal freight transport relation using machine learning algorithms. *Logistics*, 4(1):1, 2020.

DOI reference number: 10.18293/SEKE2022-0020

Enhancing Pre-Trained Language Representations Based on Contrastive Learning for Unsupervised Keyphrase Extraction

Zhaohui Wang[12], Xinghua Zhang[12], Yanzeng Li[3], Yubin Wang[12], Jiawei Sheng[12], Tingwen Liu[12*], Hongbo Xu[12]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, {wangzhaohui, liutingwen}@iie.ac.cn
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Wangxuan Institute of Computer Technology, Peking University, Beijing, China

***Abstract*—Keyphrase extraction (KPE) aims to obtain a set of phrases from a document that can summarize the main content of the document. Recently, pre-trained language models (LMs), especially BERT and ELMo, have achieved remarkable success, presenting new state-of-the-art results in unsupervised KPE. However, current pre-trained LMs focus on building language modeling objectives to learn a general representation, ignoring the keyphrase-related knowledge. Intuitively, the joint embedding of the keyphrase set should tend to be close to that of the extracted document, and far from those of other documents. In this work, we propose a contrastive learning-based semantic representation task to further improve BERT for unsupervised KPE. Particularly, we design a doc-phrase attention module to generate joint semantic embedding of the keyphrase set as a positive sample and select other semantically similar documents as hard negative samples. In the prediction layer, we further add an accumulated self-attention module to calculate the final scores of candidate phrases. We compare with eight strong baselines, and evaluate our model on three publicly available datasets. Experimental results show that our model is effective and robust on both long and short documents.**

***Index Terms*—keyphrase extraction, contrastive learning, pre-trained language models, unsupervised, attention**

I. Introduction

With a vast amount of scientific or non-scientific articles published online every year, indexing and information retrieval have become challenging. Keyphrase extraction (KPE) is concerned with automatically extracting a set of representative phrases from a document that concisely summarizes its content [1]. It can significantly accelerate the speed of information retrieval and help people get first-hand information from a long text quickly and accurately. As a result, automatic keyphrase extraction is crucial in natural language processing (NLP).

Recently, pre-trained language models (LMs), such as ELMo [2] and BERT [3], have caused a stir in the KPE community. These LMs are pre-trained on unlabeled text and then applied to KPE, in either an embedding-based unsupervised [4], [5] or a supervised [6] manner, both offering substantial performance boosts. Despite refreshing the state-of-the-art performance of KPE, the current pre-trained techniques are not directly optimized for KPE. Typically, these models build unsupervised training objectives to capture dependency between words and learn a general language representation [7], while rarely considering incorporating keyphrase information which can provide rich knowledge for KPE. Due to little knowledge connection between KPE and general language modeling, how to adapt public pre-trained models to be KPE-specific remains an open problem.

The embedding-based unsupervised KPE, which has been widely studied, ensures high retrieval speed and outstanding performance on certain datasets without a large amount of annotated data. However, these methods commonly include the following steps: extracting noun phrases from the document as candidate phrases, utilizing a pre-trained language model to generate document and phrase embeddings, calculating each candidate's final score independently. During the scoring stage, they generally use cosine similarity to assess the relevance between a single candidate phrase and document. However, the semantic information contained in the keyphrase set, which expresses the main content of a document, is ignored.

In this paper, we aim to fully utilize the joint semantic information of the keyphrase set. Inspired by the success of contrastive learning in computer vision (CV), the most recent methods are interested in determining whether it could also assist language models in promoting representation ability [8]. With the simple intuition that the joint semantics of the keyphrase set tends be close to its document and be dissimilar to other document in semantic space, we propose a contrastive learning-based semantic representation task, which leverages triplet loss [9] to effectively optimize the representations of the pre-trained language model.

Specifically, we use BERT [3] language model to encode documents and candidate phrases. In triplet loss, we use the document embeddings as anchors. The doc-phrase attention module is designed to distinguish keyphrases and non-keyphrases and generate joint semantic embeddings of keyphrase sets as positive samples. We also select semantically similar documents as hard negative samples. In the prediction stage, we utilize the linear integration of doc-phrase attention and accumulated self-attention modules to calculate the final scores of candidate keyphrases.

We compare our model with eight unsupervised keyphrase

*Corresponding author
DOI reference number: 10.18293/SEKE2022-131

extraction methods on three benchmark datasets. Two datasets contain short documents, and one contains long documents. Experimental results show that our model performs better than or as competitive as the baselines. The main contributions of this paper are summarized as follows:

- We propose a contrastive learning-based semantic representation task to enhance pre-trained LMs for unsupervised KPE;
- We design a doc-phrase attention module to generate joint semantic embedding contained in keyphrase set and combine accumulated self-attention module to calculate the scores for candidate phrases;
- Experimental results show that our model outperforms eight strong baselines and is robust to identify keyphrases from both short and long documents of different domains.

II. Related Work

This section briefly describes prior works about unsupervised keyphrase extraction and contrastive self-supervised learning, which have strong connections with this paper.

A. Unsupervised Keyphrase Extraction

Keyphrase extraction is an important problem in NLP area. Researchers have developed a wide range of solutions for this task in the last few years, including supervised and unsupervised methods. In the supervised setting, keyphrase extraction is usually treated as a classification problem [10], [11] or text generation [12], [13] task, which needs large amounts of annotated data for training and are generally domain-specific. In this paper, we discuss unsupervised keyprhase extraction only.

Traditional unsupervised methods such as TF-IDF and YAKE! [14], are statistic-based methods. YAKE! incorporates five different features for each term to calculate a ranking score after preprocessing the text by splitting it into individual terms.

In addition, graph-based methods via converting a document into graphs are popular. Motivated by Brin and Page [15], TextRank [16] was proposed to rank nodes of graphs constructed by word co-occurrence windows and implements PageRank iteratively. After this, various works attempted to expand TextRank. SingleRank [17] is one of the modifications in which the weight of each edge is equal to the number of co-occurrences of two corresponding words. TopicRank [18] assigned a salience score to each topic by candidate keyphrase clustering.

Embedding-based methods rely on notable new developments in text representation learning by encoding text sequences into low-dimension vectors. Hence, embedding-based unsupervised keyphrase extraction has gained a lot of attention in recent years. EmbedRank [19] proposed to measure the text similarity between phrase and document embeddings to make predictions. Sun et al. [4] proposed SIFRank, which improves the static embedding of EmbedRank with a pre-trained language model and a sentence embedding model SIF [20]. AttentionRank [5] proposed a hybrid attention model to identify keyphrases from a document. These embedding-based methods ignore the information carried in the keyphrase set, while we effectively capture and utilize the information by a doc-phrase attention module and achieve competitive results.

B. Contrastive Self-Supervised Learning

Self-supervised learning has gained popularity due to its ability to avoid the cost of annotated large-scale datasets. It can adopt self-defined pseudo labels as supervision and use the learned representations for several downstream tasks. Specifically, contrastive learning has recently become a dominant component in self-supervised learning methods in CV, NLP, and other domains [21]. Contrastive learning aims to learn effective representation by pulling semantically close neighbors together and pushing non-neighbors apart. CERT [8] applied the back-translation to create augmentations for original sentences. Declutr [22] regarded that different spans inside one document are similar to each others. SCL [23] proposed a supervised contrastive learning objective to increase the distance between categories for the fine-tuning stage. SimCSE [24] described an unsupervised approach, which takes an input sentence and predicts itself in a contrastive objective, with only standard dropout used as noise. Intuitively, the joint semantics of the keyphrase set tends to be close to the entire document, and far from other documents in semantic space. Therefore, we desingn a contrastive learning-based semantic representation task to enhance pre-trained language model for unsupervised KPE.

III. Problem Definition

Keyphrase extraction is the task of automatically selecting a small set of phrases that summarize the document's main content. Formally, given a document $d = \{w_1, w_2, \ldots, w_n\}$ in dataset D consisting of n words, candidate phrases can be selected as set $C = \{c_1, c_2, \ldots, c_m\}$, where m is the number of candidates. Each candidate c_i consists of several words $c_i = \{c_i^1, c_i^2, \ldots, c_i^l\}$. Keyphrase extraction is to select Top-K candidates from C forming a keyphrase set $K = \{k_1, k_2, \ldots, k_t\}$ according to their scores, usually $t < m$.

IV. Methodology

In this section, we introduce our method in detail. The overall architecture is illustrated in Figure 1, which can be divided into three parts: (a) extracting a candidate set C from a document for the contrastive framework and prediction stage; (b) our contrastive architecture to enhance BERT model and (c) the final score calculation strategy for each candidate.

A. Candidate Generation

We use the candidate generation module implemented in EmbedRank [19]. Firstly, the document is tagged to a sequence of words with part-of-speech tags. Then, we extract the noun phrases (NPs) from the sequence according to the part-of-speech tags using NP-chunker (pattern written by regular

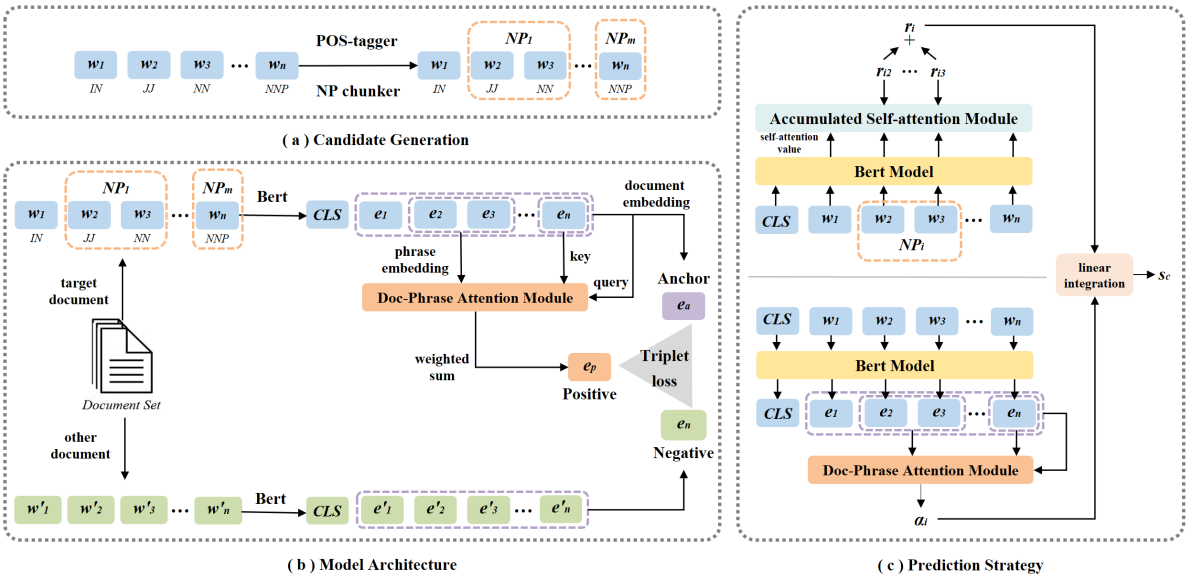


Fig. 1. Overview of Our Method.

expression). The NPs extracted from the document are the candidate keyphrases. Specifically, we use the Stanford CoreNLP to tokenize part-of-speech tags. And regular expression <NN.*|JJ>*<NN.*> is used to extract noun phrases as candidate keyphrases.

B. Model Architecture

Existing embedding-based works measure the semantic similarity between each candidate phrase and the document independently, which ignores the semantic information contained in the keyphrase set. Intuitively, the joint semantics of the keyphrase set, which can better describe the main content of a document, tends to be close to the entire document, but far from other documents in semantic space. In this part, we introduce how to generate a joint semantic embedding from a phrase set and how to model the relationship between keyphrase set and document embeddings using ontrastive learning.

1) Encoder: BERT model encodes document d into a sequence of vectors $E_d = \{e_d^1, e_d^2, e_d^3, \ldots, e_d^n\}$, where e_i^d is the i-th word's contextualized embedding in document d from the last transformer layer.

$$E = \text{Bert}\left(w_1, w_2 \ldots, w_n\right) \tag{1}$$

MeanPooling method is adopted to obtain document-level and phrase-level embeddings.

$$e_d = \text{MeanPooling}\left(e_d^1, e_d^2, e_d^3, \ldots, e_d^n\right) \tag{2}$$

$$c_i = \text{MeanPooling}\left(c_i^1, c_i^2, \ldots, c_i^l\right) \tag{3}$$

2) Doc-Phrase Attention Module: The doc-phrase attention is designed to measure the importance between candidate phrases. Considering that only part of the candidate phrases are keyphrases, we design a doc-phrase attention module to distinguish keyphrases from non-keyphrases. The input of doc-phrase attention module is the embedding of document d and the embeddings of candidate phrases in set C, where the former is used as query and the latter is considered as key and value. The joint semantic embedding of phrase set for target document d can be calculated as the weighted sum of c_i.

$$\text{Att}\left(e_d, c_i\right) = e_d^T W c_i \tag{4}$$

$$\alpha_i = \frac{e^{\text{Att}(e_d, c_i)}}{\sum_{i=1}^{m} e^{\text{Att}(e_d, c_i)}} \tag{5}$$

$$e_t = \sum_{i=1}^{m} \alpha_i c_i \tag{6}$$

3) Contrastive supervision: Triplet loss [9] was used as the overall training objective. In order to ensure fast convergence, we design an effective strategy to select hard negative samples.

Given the triplets (e_a, e_p, e_n) where a, p, n represent anchor, positive and negative examples respectively, triplet loss aims to narrow the gap between anchor and positive examples and distinguish between anchor and negative examples. The constraint is shown as Equation 7. Typically, a document's main content tends to be close to itself and far from other documents in semantic space. We take the target document embedding as the anchor, the joint semantic embedding of the target document as the positive sample and the embedding of other documents in dataset as the negative samples.

$$\left\|e_a - e_p\right\|_2^2 + m < \left\|e_a - e_n\right\|_2^2, \forall \left(e_a, e_p, e_n\right) \in \mathbb{R} \tag{7}$$

The loss function can be defined as the following:

$$L = \sum_{d \in D} \left[\left\|e_d^a - e_d^p\right\|_2^2 - \left\|e_d^a - e_d^n\right\|_2^2 + m\right]_+ \tag{8}$$

https://stanfordnlp.github.io/CoreNLP/

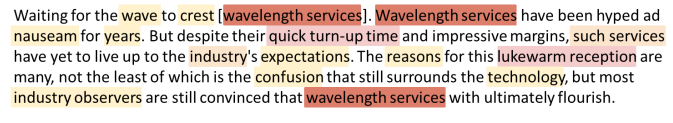


Fig. 2. Visualization Example of The Accumulated Self-attention Module.

where e_d^a and e_d^n represent the embedding of target document and the embedding of other documents calculated by Equation 2. Furthermore, e_d^p represents the joint semantic embedding of target document's phrase set calculated by Equation 6 and m denotes margin.

In addition, it is crucial to select or mining triplets that violate the triplet constraint in Equation 7. This means that, given e_a we need select an e_p (*hard positive*) such that $\text{argmax}_{e_p} \|e_a - e_p\|_2^2$ and similarly e_n (*hard negative*) such that $\text{argmax}_{e_n} \|e_a - e_n\|_2^2$. We select documents that are semantically closer to the target document as negative examples. As for the positive example, according to experimental results, the joint semantic embedding generated by doc-phrase attention module can meet the triplet loss requirements.

C. Prediction Strategy

1) Accumulated Self-attention Module: Motivated by [5], [25], we extract self-attention weights of the words from the BERT. As shown in Figure 2, we sum the attention weights that a phrase received in the document, and all the noun phrases are highlighted. The higher self-attention it receives, the darker the noun chunk is. Intuitively, noun phrases with darker colors should be selected as keywords with higher probabilities. The calculating method is introduced as follows.

To obtain the attention value r_w of the word w within a sentence, we sum the attentions $r_{w'w}$ that a word w received from other words w' within the same sentence s, shown as Equation 9. This attention value r_w represents the importance of the word within the context of a sentence.

$$r_w = \sum_{w' \in s} r_{w'w} \tag{9}$$

To calculate the self-attention of a candidate c in sentence j, we add up the attentions of the words in c, shown as Equation 10.

$$r_j^c = \sum_{w \in c} r_w \tag{10}$$

The document level self-attention value of candidate c is computed as the sum of all self-attention values of c in each sentence of document d, shown as Equation 11.

$$r_c = \sum_{j \in d} r_j^c \tag{11}$$

2) Final Score Calculation: For document d, the doc-phrase attention value α_c and the accumulated self-attention value r_c are calculated and normalized separately for each candidates. The final score of a candidate is generated by linear integration of these two values using Equation 12, where $\beta \in [0, 1]$.

$$S_c = \beta * r_c + (1 - \beta) * \alpha_c \tag{12}$$

TABLE I
STATISTICS OF THE THREE DATASETS.

Dataset	Documents				Keyphrases		
	Total	Type	AveWords	AveSentences	Total	AveNumber	AveLength
Inspec	500	Abstracts	134	6	4912	9.8	2.3
SemEval2017	493	Paragraph	168	7	8529	17.3	3
SemEval2010	243	Full papers	8154	369	3662	15.1	2.1

V. EXPERIMENTS

In this section, we first set up the experiments by preparing the datasets and introducing the comparison methods, and then report the results of conducted experiments to demonstrate the effectiveness of the proposed method.

A. Datasets and Evaluation Metrics

To fully evaluate the performance of our model, we testify it on three benchmark datasets. The statistics of the three datasets are shown in Table I. Datasets Inspec [26] and SemEval2017 [27] contain short documents, whereas SemEval2010 [28] contains long documents.

The **Inspec** dataset consists of 2000 short documents selected from scientific journal abstracts. There are 1000 documents for training, 500 for validation and 500 for test. We use the test part to validate our model in this paper.

The **SemEval2017** dataset is the Task 10 in SemEval2017 competition. It contains 493 paragraphs selected from ScienceDirect journal, covering computer science, materials science and physics. Each document is annotated with keyphrases by an undergraduate and an expert.

The **SemEval2010** dataset consists of 243 full papers from the ACM Digital Library. The articles are purposefully selected from four different areas.

B. Baselines

We compared our model with eight keyphrase extraction methods which are all unsupervised models in three types: statistic-based model, graph-based model and embedding-based model. The statistic-based models are TF-IDF and YAKE! [14]. The graph-based models are TopicRank [18], PositionRank [29] and SingleRank [17]. The embedding-based models are EmbedRank [19], SIFRank [4] and AttentionRank [5].These baselines all generate candidates using noun phrases without any additional steps. We used PKE to run SingleRank, RAKE, and TopicRank. The published GitHub code of YAKE!, PositionRank, EmbedRank, SIFRank and AttentionRank were used to produce the results on the selected

https://github.com/boudinfl/pke
https://github.com/LIAAD/yake
https://github.com/ymym3412/position-rank
https://github.com/swisscom/ai-research-keyphraseextraction
https://github.com/sunyilgdx/SIFRank
https://github.com/hd10-iupui/AttentionRank

TABLE II
MODEL COMPARISION WITH PRECISION(P), RECALL(R), AND F-SCORE(F1) @5, @10, @15 ON THREE BENCHMARK DATASETS. N IS THE NUMBER EXTRACTED FROM A SINGLE DOCUMENT BY THE MODELS. THE BEST PERFORMANCES ARE BOLD.

N	Method	Inspec			SemEval2017			SemEval2010		
		P	R	F1	P	R	F1	P	R	F1
5	**TF-IDF**	16.71	8.51	11.28	28.31	8.18	12.69	14.93	4.72	7.17
	YAKE!	25.04	11	15.29	24.79	7.96	12.05	16.87	5.65	8.46
	TopicRank	27.4	11.93	16.62	38.13	11.02	17.1	10.37	3.52	5.26
	PositionRank	29.8	12.15	17.26	40.65	11.75	18.23	5.16	1.62	2.47
	SingleRank	30.26	12.24	17.43	40.57	12.6	19.23	2.33	1.41	1.76
	EmbedRank	33.77	12.43	18.17	44.72	12.93	20.06	3.29	1.1	1.65
	AttentionRank	35.44	12.72	18.72	45.27	13.15	20.38	19.51	6.3	9.52
	SIFRank	**39.64**	13.83	20.51	45.16	13.23	20.46	11.44	3.83	5.74
	Ours	39.04	**14.01**	**20.61**	**47.30**	**13.74**	**21.30**	**22.22**	**7.18**	**10.85**
10	**TF-IDF**	13.76	14.01	13.88	22.19	12.83	16.26	13.18	9.59	11.1
	YAKE!	19.48	16.67	17.97	23.33	14.86	18.16	14.94	10	11.98
	TopicRank	27.11	22.27	24.45	30.87	17.84	22.61	9.26	6.2	7.43
	PositionRank	28.04	23.25	25.42	33.1	20.2	25.09	4.61	3.05	3.67
	SingleRank	28.32	23.43	25.64	35.25	20.38	25.83	2.23	2.53	2.37
	EmbedRank	29.97	22.3	25.57	37.48	23.21	28.67	3.58	2.34	2.83
	AttentionRank	31.47	23.15	26.68	39.66	23.03	29.14	16.83	10.87	13.21
	SIFRank	**35.89**	**24.77**	**29.31**	40.31	23.32	29.55	7.82	5.18	6.23
	Ours	34.06	24.25	28.33	**41.52**	**24.12**	**30.52**	**19.18**	**12.39**	**15.05**
15	**TF-IDF**	11.44	17.47	13.83	18.01	15.62	16.73	12.16	11.39	11.76
	YAKE!	17.12	21.7	19.14	21.41	20.07	20.72	12.87	12.86	12.86
	TopicRank	24.09	29.04	26.33	26.85	23.17	24.87	7.98	8.06	8.02
	PositionRank	24.59	28.53	26.41	29	26.5	27.69	4.15	4.02	4.08
	SingleRank	27.34	27.26	27.3	32.95	28.48	30.55	2.62	4.37	3.28
	EmbedRank	26.41	30.2	28.18	34.68	29.61	31.95	3.65	3.63	3.64
	AttentionRank	28.43	29.09	28.76	35.26	30.64	32.79	14.24	13.79	14.01
	SIFRank	**30.84**	**30.98**	**30.91**	35.90	31.10	33.33	6.20	6.11	6.15
	Ours	29.53	30.57	30.04	**37.05**	**32.25**	**34.48**	**16.27**	**15.76**	**16.01**

datasets. It is worth noting that the produced results of the baselines are slightly higher or lower than the results presented in the original papers.

C. Experiment Settings

In the experiment, we use "bert-base-uncased" as our pre-trained model. Our model is optimized using Adam with 1e-5 learning rate and 8 batch sizes. We use maximum sequence length 512. The number of negative samples is 2. For all datasets, we set the linear combination ratio β to be 0.5 for Inspec and 0.8 for SemEval2017 and SemEval2010. For the baseline methods, the parameters published on the corresponding GitHub were used. All of the models are implemented under PyTorch running on 2 NVIDIA Tesla T4 GPUs.

D. Results

Table II shows the results of Precision, Recall and F1 @5, 10, and 15 using our model and baseline models on three datasets.

Short document. The results show that the embedding-based methods, including our model, perform better than the statistic-based (TF-IDF and YAKE!) and graph-based algorithms (SingleRank, TopicRank, and PositionRank) on short document sets (Inspec and SemEval 2017). Statistic-based and graph-based unsupervised methods, despite their simplicity, do not perform as well as other methods on short documents, for which semantic information is assumed to be very important.

SIFRank performs slightly better than our model on Inspec. It works better than our model when K is set to 10 or 15.

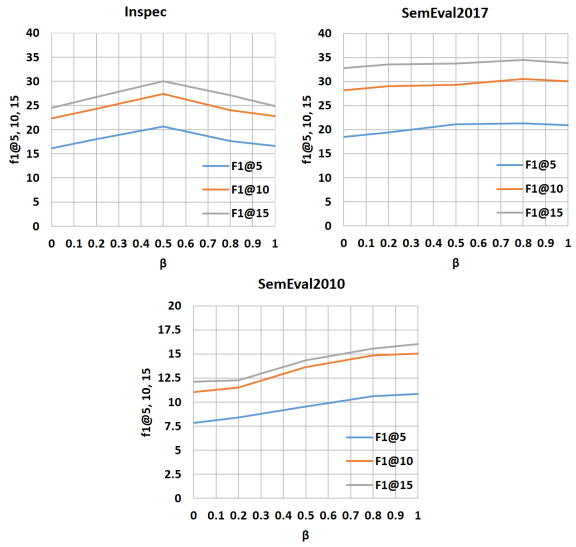


Fig. 3. Evaluation of the Linear Proportions' Impact on Performance.

Nevertheless, our model has a slightly better F1 than SIFRank and other baselines when the top 5 candidates are used for evaluation. Moreover, our model outperforms SIFRank on SemEval2017 dataset. It shows that our method performs competitively with SIFRank.

Long document. Our method shows advantage on long document set SemEval2010. The F1 value is at least 1.3% better than the highest baseline. Statistic-based methods achieve prominent results than other baselines on long documents. This may indicate that for long documents, statistical features such as word frequency and inverse document frequency are more important for the selection of keyphrases. Existing graph-based methods and embedding-based methods have difficulty in capturing these features, which can be well solved in our method.

E. Impact of hyperparameters

Our model linearly integrates the doc-phrase attention value and the accumulated self-attention value to measure the importance of a candidate phrase. We study the influence of the two modules by adjusting β (in Equation 12) from 0 to 1. Figure 3 shows that the best ratio is different for different datasets.

For short document datasets such as Inspec and SemEval2017, the addition of both parts of the attention values can improve the model performance. Specifically, for dataset Inspec, F1 value is highest when β is round 0.5. For SemEval2017, the best performance can be achieved when β is set to 0.5, 0.8 and 0.8. However, the contribution of accumulated self-attention value is higher than doc-phrase attention value for long ducument dataset-SemEval2010. When β is set to 1, the model achieves the best performance, which means only accumulated self-attention value is needed to find the keyphrases.

We consider that the accumulated self-attention module captures the repetition of the keyphrases implicitly through

the self-attention weights accumulation over the document. However, for short document dataset like Inspec, the doc-phrase attention value has more impact. Since there are only a few sentences in a document, the repetition of the phrases is low. Nonetheless, the contextual relevance among keyphrases and sentences and documents still needs to be emphasized.

VI. Conclusion

This paper proposes a contrastive learning-based semantic representation task to enhance pre-trained language model for unsupervised keyphrase extraction, which takes advantage of triple loss to combine the target document, joint information of keyphrase set, and other documents in semantic space. We utilize a doc-phrase attention module and an accumulated self-attention module to rank candidate phrases. The doc-phrase attention is designed to measure the importance between candidate phrases. The accumulated self-attention module aims to determine the importance of a candidate phrase in the context of the document. We compared the proposed model with eight strong baselines on three benchmark datasets, including two short document datasets and one long document dataset. Our model gains a better or competitive F1@5, 10, and 15 on all datasets. The ablation study shows that accumulated self-attention has a higher contribution to the long document set. The linear integration of the two attention modules shows the best results for short documents. In conclusion, our model is an efficient and robust unsupervised method for keyphrase extraction task, which regards the keyphrase set as a whole and fully leverages the semantic information from keyphrase set and document.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (grant No.2021YFB3100600), the Strategic Priority Research Program of Chinese Academy of Sciences (grant No.XDC02040400) and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

References

[1] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 1262–1273.

[2] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Jun. 2018, pp. 2227–2237.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] Y. Sun, H. Qiu, Y. Zheng, Z. Wang, and C. Zhang, "Sifrank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model," *IEEE Access*, vol. 8, pp. 10 896–10 906, 2020.

[5] H. Ding and X. Luo, "Attentionrank: Unsupervised keyphrase extraction using self and cross attentions," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 1919–1928.

[6] S. Sun, Z. Liu, C. Xiong, Z. Liu, and J. Bao, "Capturing global informativeness in open domain keyphrase extraction," in *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 2021, pp. 275–287.

[7] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang, and F. Wu, "Skep: Sentiment knowledge enhanced pre-training for sentiment analysis," *arXiv preprint arXiv:2005.05635*, 2020.

[8] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, "Cert: Contrastive self-supervised learning for language understanding," *arXiv preprint arXiv:2005.12766*, 2020.

[9] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[10] R. Alzaidy, C. Caragea, and C. L. Giles, "Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents," in *The world wide web conference*, 2019, pp. 2551–2557.

[11] D. Sahrawat, D. Mahata, M. Kulkarni, H. Zhang, R. Gosangi, A. Stent, A. Sharma, Y. Kumar, R. R. Shah, and R. Zimmermann, "Keyphrase extraction from scholarly articles as sequence labeling using contextualized embeddings," *arXiv preprint arXiv:1910.08840*, 2019.

[12] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," *arXiv preprint arXiv:1704.06879*, 2017.

[13] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *arXiv preprint arXiv:1603.06393*, 2016.

[14] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, "Yake! collection-independent automatic keyword extractor," in *European Conference on Information Retrieval*. Springer, 2018, pp. 806–810.

[15] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[16] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[17] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge." in *AAAI*, vol. 8, 2008, pp. 855–860.

[18] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *International joint conference on natural language processing (IJCNLP)*, 2013, pp. 543–551.

[19] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," *arXiv preprint arXiv:1801.04470*, 2018.

[20] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *International conference on learning representations*, 2017.

[21] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021.

[22] J. M. Giorgi, O. Nitski, G. D. Bader, and B. Wang, "Declutr: Deep contrastive learning for unsupervised textual representations," *arXiv preprint arXiv:2006.03659*, 2020.

[23] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," *arXiv preprint arXiv:2011.01403*, 2020.

[24] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[25] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," *arXiv preprint arXiv:1906.04341*, 2019.

[26] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 2003, pp. 216–223.

[27] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, "Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications," *arXiv preprint arXiv:1704.02853*, 2017.

[28] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010, pp. 21–26.

[29] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1105–1115.

Exploring Relevance and Coherence for Automated Text Scoring using Multi-task Learning

Yupin Yang
College of Computer Science
Chongqing University
Chongqing, China
yyp@cqu.edu.cn

Jiang Zhong
College of Computer Science
Chongqing University
Chongqing, China
zhongjiang@cqu.edu.cn

Chen Wang
College of Computer Science
Chongqing University
Chongqing, China
chenwang@cqu.edu.cn

Qing Li
School of computer science
Northwestern Polytechnical University
Xi'an, China
qingli@nwpu.edu.cn

***Abstract*—With the explosive growth of the information on the Internet, the evaluation of the quality and credibility of web content has become more important than ever before. In this work, we focus on the quality assessment of texts. Recently, various methods have been proposed for the automated text scoring task and obtained competitive results. However, few studies have focused on both relevance and coherence, which are two important factors in evaluating text quality. To improve the scoring task, we propose two auxiliary tasks using negative sampling and integrate them into a multi-task learning framework. The first auxiliary task is relevance modeling and the other one is coherence modeling. We evaluate our model on the Automated Student Assessment Prize (ASAP) dataset. Experimental results show that our model achieves higher Quadratic Weighted Kappa (QWK) scores with an improvement of 1.5% on average.**

***Keywords*—automated essay scoring, multi-task learning, natural language processing**

I. Introduction

Web2.0 accelerates the transition to Web3.0, and global data storage presents explosive growth. The Internet is flooded with all kinds of information. Organizations with poor website quality or inefficient service may establish a bad image and weaken the status of the organization. Therefore, it is necessary to develop effective Web content quality control. Research shows that if visitors find the site pleasant, they are more likely to visit the site again. Accordingly, we explore utilizing neural network models to assess the quality of texts.

Automated text scoring (ATS) aims to predict scores related to the quality of a text. Evaluating texts is time-consuming, and different evaluators may grade different scores to the same text. In this case, the computer-based automatic text scoring system can effectively overcome the inadequacies of manual scoring [1]. Typically, researchers use a combination of Natural Language Processing (NLP) and machine learning to perform this task.

DOI reference number: 10.18293/SEKE2022-024

Existing ATS models can be divided into two types: feature engineering-based and neural networks-based methods. The approach based on feature engineering uses handcrafted features (e.g., text length) to score texts. For example, the Enhanced AI Scoring Engine (EASE) is a typical model that has been shown to work well [2]. These models are highly interpretable but require additional engineering. To address the issues, the neural networks-based approach automatically extracts features (e.g., lexical features) from texts for the final grading. Recently, these approaches have achieved high performance. For example, Taghipour and Ng [3] innovatively used Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) to learn text features. Dong et al. [4] adopted Recurrent Convolutional Neural Network (RCNN) with an attention mechanism to model this task and grade the texts automatically.

The automated text scoring task is usually evaluated on the ASAP dataset. However, few studies have focused on the relevance of essays and prompts as well as the coherence of sentences in the essay. Note that the prompt refers to the topic of the essay, which usually consists of reading materials and task descriptions. Relevance is how an essay fits the topic and coherence is what makes multi-sentences of text logical and syntactical. To capture the relevance between the essay content and the prompt, Chen et al. [5] incorporated the similarity between the essay and the prompt into the final representation to grade the text. To further obtain the relevance between each sentence and the source materials, Zhang et al. [6] introduced the co-attention based neural network to model the similarity between them. Besides, the coherence of sentences in the essay is also one of the important criteria for scoring. To our knowledge, a well-written essay is more coherent than a random combination of sentences. Based on this point, Mesgar et al. [7] introduced a local coherence model to obtain the flow of content that semantically connects adjacent sentences in the essay. Li et al. [8] employed the self-attention mechanism to learn the relationship between long-distance words in the essay

to estimate coherence scores.

In this work, we propose a multi-task learning framework for automated text scoring. In our framework, two auxiliary tasks are introduced including relevance modeling task and coherence modeling task. The relevance modeling task aims to enhance the ability to extract prompt-specific features. Specifically, we mix all the essays under different prompts together and feed them into the model, and then predict which prompt the essay belongs to. The coherence modeling task aims at enhancing the ability to capture the discourse coherence of essays. In our model, this task can be regarded as a binary classification task. We randomly select three consecutive sentences as a group and make some modifications to the group-based data to construct negative samples, and then predict whether the essay is coherent or not. Finally, we integrate these two auxiliary tasks with the scoring task into a multi-task learning framework for final scoring.

To verify the effectiveness of our multi-task learning framework, we conduct experiments on the ASAP dataset. Experimental results show that our method achieves better performance than previous methods, which demonstrates that our proposed method is effective for automated text scoring. Our work also shows that auxiliary tasks can enhance the performance of the BERT model on downstream tasks.

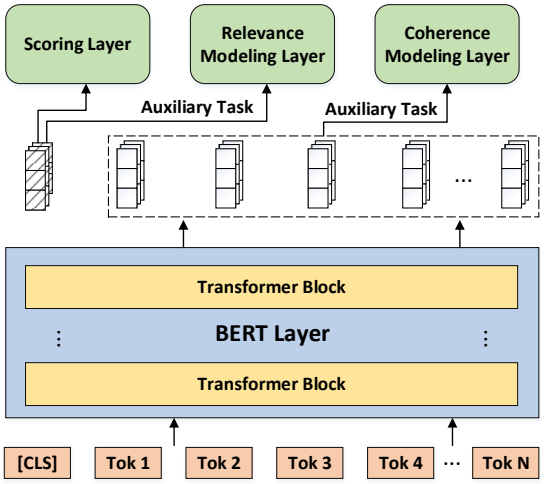


Figure. 1. An overview of our multi-task learning architecture. It first adopts BERT as the shared encoder. Then, three task-specific layers are connected behind the encoder, sharing the text representation learned from the BERT layer.

II. Related Work

The discussion of related work is divided into two subsections: ATS-related and MTL-related. In our model, we apply multi-task learning to the task of automated text scoring. There is a long history of automated essay scoring and multi-task learning, and in this chapter, we concisely review some common methods.

A. Automated Essay Scoring

Automated essay scoring is an important application of natural language processing (NLP) in education. Previous methods were mainly based on feature engineering, in which the ATS task was considered as a classification or regression problem. In the case of the former, the classifier directly outputs the label that represents the score. In the latter case, the output is in the range of the golden score. The feature-engineering methods require handcrafted features, which include some statistical features such as essay length, number of spelling errors, etc. These approaches include e-rater [9], PEG [10], and EASE [2]. In the PEG, more than thirty writing quality factors are considered. Besides, Cozma [11] combined string kernels and word embeddings to capture text features, namely the bag-of-super-word-embeddings (BOSWE). Dascalu et al. [12] implemented an automated essay scoring system for Dutch by integrating features such as lexical and semantics features.

To avoid the need for feature engineering, researchers begin to explore the application of neural networks to the automated essay scoring task. Taghipour and Ng [3] innovatively combined CNN and Long Short-Term Memory (LSTM) to learn text representations for final scoring and obtained competitive results. Dong et al. [4] adopted ConvNet and LSTM to learn sentence representation and text representation respectively. Mesgar et al. [7] employed the RNN layer for the words in the sentence to integrate contextual information. Yang et al. [13] introduced the BERT model to learn text representations.

B. Multi-task Learning

Multi-task learning was firstly proposed in 1994 by Caruana [14] to improve the generalization ability of the model. It is an inductive transfer mechanism that shares parameter information between multiple tasks [15]. Compared with single-task learning, multi-task learning refers to learning multiple tasks simultaneously, which can achieve better performance. In multi-task learning, the main task and auxiliary tasks learn from each other and jointly enhance the generalization ability [16]. For auxiliary tasks, a basic assumption is that auxiliary tasks should be related to the main task and can promote the learning of the main task.

Multi-task learning has been widely used across applications of machine learning, from natural language processing [17] and speech recognition [18] to computer vision [19]. Liu et al. [20] introduced Two-Stage Learning Framework for ATS where semantic score, coherence score, and prompt-relevant score are computed at the first stage and they are combined with handcrafted features in the second stage. Nadeem et al. [21] used natural language inference and discourse marker prediction as auxiliary tasks for capturing discourse characteristics of essays.

III. Method

In this section, we demonstrate the main steps of our proposed model. It consists of a shared encoder and three task-specific layers including the scoring task, relevance modeling

task, and coherence modeling task. The overview of our proposed model is shown in Fig.1. In the following subsections, more details of each module will be introduced.

A. Shared Encoder

Large pre-trained language models (e.g., BERT [22], GPT [23], and XLNet [24]) have shown the remarkable ability of representation and generalization in many tasks. These pre-trained language models achieve great success in learning text representations with deep semantics. In our framework, we choose BERT as the shared encoder to better capture the semantics of the given essay.

BERT is trained on enormous corpora with more than 3000M words. It has two target tasks, including the masked language model and next sentence prediction. Many NLP downstream tasks, such as sentence classification and question answering, have gained benefits by utilizing pre-trained BERT to learn text representation. Specifically, we adopt RoBERTa [25] as the encoder to get better performance. The self-attention mechanism [26] is the key to the success of BERT, in which a sequence calculates the word weights with itself. The attention process is defined as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where Q, K, and V are the matrix transformed from the input. d_k denotes the hidden dimension of matrix K.

Given an input essay $E = \{w_1, w_2, ..., w_N\}$, where N is the length of the essay, we add a special token [CLS] at the beginning of the sequence and get a new sequence $E' = \{[CLS], w_1, w_2, ..., w_N\}$. Then, we send the sequence E' into a pre-trained BERT and take the hidden state of the [CLS] token as the text representation:

$$H = BERT(E') \tag{2}$$

$$R = H_{[CLS]} \tag{3}$$

where H denotes the final hidden state sequence of BERT, and R is the text representation.

B. The Scoring Task

The main task of our method is the scoring task, which aims to predict a score for each essay. Following other ATS methods [3], [20], [21], [27], we utilize a dense layer to compute the score with the text representation R as:

$$p_s = sigmoid(W_s R + b_s) \tag{4}$$

where W_s is the weight matrix, b_s is the bias initialized with the mean score of the essays, and sigmoid is the activation function to normalize the calculated score into [0,1]. In this part, we select mean square error (MSE) as the loss function:

$$L_S = \frac{1}{m}\sum_{i=1}^{m}(y_s - p_s)^2 \tag{5}$$

where y_s is the true target, and p_s is the prediction result for the scoring task.

C. Relevance Modeling Task

To our knowledge, there is a close relevance between the essay content and the topics. ATS systems may give a high score to an unrelated but well-written essay. However, a human rater will give higher scores to those essays related to the topics, and lower scores to those essays that are not relevant to the topics. To exploit the prompt-specific knowledge, we design this auxiliary task named relevance modeling. Since high-scoring essays always stick to the prompt, we first mix up the top 40% essays of all prompts, and their labels are the prompt they belong to. After that, we feed the latent text representation R learned from BERT into a dense layer to predict the prompt:

$$p_{rm} = softmax(W_{rm} R + b_{rm}) \tag{6}$$

where W_{rm} is the weight matrix, b_{rm} is the bias, and softmax is the activation function for the multi-classifier. In this module, we optimize this auxiliary task with the cross-entropy loss as:

$$L_{RM} = -\sum y_{rm} * log(p_{rm}) \tag{7}$$

where y_{rm} is the true target, and p_{rm} is the prediction result.

D. Coherence Modeling Task

Language learners may have learned to make both meaningful and grammatical sentences but do not know how to organize the sentences together to construct a good essay. Coherence is what makes a multi-sentence text meaningful, both logically and syntactically. In this work, our coherence modeling task is used to better capture the discourse coherence of essays.

In this task, we regard the coherence modeling task as a binary classification task. The vanilla essay is with the label "1" while the others are with the label "0". To construct negative samples, we perform three operations on the essays. Note that each negative sample is prompt-related but incoherent. For an input essay, we randomly select three consecutive sentences as a group. There are three ways to construct negative samples: 1) delete operation—removing the selected group; 2) replace operation—replacing the selected group with a new group from another essay of the same topic; 3) inserting operation—appending a group from another essay under the same prompt to the beginning or end of the selected group. For example, given an essay E consisting of k sentences $E = \{s_1, ..., s_{i-1}, s_i, s_{i+1}, s_{i+2}, s_{i+3}, ..., s_k\}$, the one with delete operation is formed as $E^* = \{s_1, ..., s_{i-1}, s_{i+3}, ..., s_k\}$. The one with replace operation is denoted as $E^* = \{s_1, ..., s_{i-1}, s_j^{'}, s_{j+1}^{'}, s_{j+2}^{'}, s_{i+3}, ..., s_k\}$. The one with inserting operation can be $E^* = \{s_1, ..., s_{i-1}, s_i^{'}, s_{i+1}^{'}, s_{i+2}^{'}, s_i, s_{i+1}, s_{i+2}, s_{i+3}, ..., s_k\}$ or $E^* = \{s_1, ..., s_{i-1}, s_i, s_{i+1}, s_{i+2}, s_j^{'}, s_{j+1}^{'}, s_{j+2}^{'}, s_{i+3}, ..., s_k\}$. After that, we send negative samples and positive samples into the model for training. Moreover, to reduce the deviation caused by imbalanced samples, we mix the positive sample and the negative sample evenly. Our approach is based on the

TABLE I
STATISTICS OF THE ASAP DATASET.

Prompt	Essay Type	#Essays	Avg length	Scores
1	Argumentative	1,783	350	2-12
2	Argumentative	1,800	350	1-6
3	Source-Dependent	1,726	150	0-3
4	Source-Dependent	1,772	150	0–3
5	Source-Dependent	1,805	150	0–4
6	Source-Dependent	1,800	150	0–4
7	Narrative	1,569	250	0–30
8	Narrative	723	650	0–60

assumption verified in Lin et al.'s work [28] that the original article is always more coherent than the changed one.

In this module, we feed the hidden states H obtained from BERT into a Bi-LSTM network to model the semantic relationships among sentences:

$$h_t = \text{Bi-LSTM}(h_{t-1}, H_i) \tag{8}$$

where H_i is the i-th output of the BERT layer and h_t is the hidden state of the Bi-LSTM at time t. We concatenate the forward and backward output together and obtain the last hidden state h_N. Then, a fully connected layer is adopted to predict whether the essay is coherent or not:

$$p_{cm} = sigmoid(W_{cm} h_N + b_{cm}) \tag{9}$$

where W_{cm} is the weight matrix, b_{cm} is the bias, and sigmoid denotes the activation function for the binary classification task. We then train this task with the binary cross-entropy loss as:

$$L_{CM} = -y_{cm} * log(p_{cm}) - (1 - y_{cm}) * log(1 - p_{cm}) \tag{10}$$

where y_{cm} is the true target and p_{cm} is the prediction result.

While training, we alternatively optimize the scoring task with L_S in Equation (5), the relevance modeling task with L_{RM} in Equation (7), and the coherence modeling task with L_{CM} in Equation (10). The 'mix ratio' of the three tasks is set as $\lambda_S : \lambda_{RM} : \lambda_{CM} = 0.6 : 0.2 : 0.2$.

IV. EXPERIMENT

In this section, we introduce the ASAP dataset and experiment settings firstly. Then the evaluation metric is illustrated. In addition, baseline models, results of the experiment, and analyses are displayed.

A. Experiment Settings

We use a common dataset for the ATS task, which is from a Kaggle competition. There are 8 prompts of different genres, and the number of essays in the dataset is 12976. In Table I, we list some statistics of the ASAP dataset.

We implement our model using Pytorch and the BERT comes from HuggingFace [29]. Since the average length of the essay in prompt 8 is 650, we truncate the essays with the max length of 512 words. For the BERT model, we use the uncased $BERT_{base}$ model with 12 layers, 768 hidden units, and 12 heads. We use the pre-trained parameters and fine-tune the parameters with the learning rate set to 1e-5. Following previous works, we also utilize 5-fold cross-validation to evaluate our model with a 60/20/20 split for train, validation, and test sets.

B. Evaluation Metric

QWK is the official evaluation metric in the ASAP competition, which measures the agreement between ratings assigned by humans and ratings predicted by ATS systems. Following previous works, we adopt QWK as the evaluation metric. The quadratic weight matrix is calculated as follows:

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{11}$$

where i and j are gold scores and calculated scores respectively. N is the number of possible ratings. The QWK value is defined as:

$$\kappa = 1 - \frac{\sum W_{i,j} O_{i,j}}{W_{i,j} E_{i,j}} \tag{12}$$

Where O is the observed score matrix and E is the expected score matrix. $O_{i,j}$ denotes the number of essays that receive rating i by human rater and j by ATS system. E is calculated as the outer product of histogram vectors of the two (reference and hypothesis) ratings.

C. Baselines

In this section, we introduce several baseline models. Enhanced AI Scoring Engine (EASE) is a statistical model based on hand-crafted features such as length-based features and part-of-speech tags. After feature extraction, support vector regression (SVR) and bayesian linear ridge regression (BLRR) are used to build the model [2]. Cozma et al. [11] proposed HISK+BOSWE, which combined string kernels and word embeddings to extract text features on both low-level character n-gram features and high-level semantic features. Wang et al. [30] proposed RL1, which is a reinforcement learning framework incorporating quadratic weighted kappa as guidance to optimize the scoring system. Taghipour and Ng [3] proposed to assemble CNN and LSTM. Dong et al. [4] introduced hierarchical neural networks with attention mechanisms to learn the representation of essays. Tay et al. [31] proposed SKIPFLOW LSTM, where there is a mechanism to simulate the relationship between hidden states in the LSTM network during reading, so as to learn the characteristics of text coherence. Yang et al. [13] proposed R^2BERT that utilized a pre-trained language model to get the scores and used mean square error loss and the batch-wise ListNet loss with dynamic weights to constrain the scores simultaneously.

https://www.kaggle.com/c/asap-aes/data

https://github.com/huggingface/transformers

https://github.com/edx/ease

TABLE II
QWK SCORES OF DIFFERENT METHODS ON THE ASAP DATASET (* DENOTES STATISTICAL MODEL). IN THIS TABLE, RM DENOTES THE RELEVANCE MODELING TASK AND CM DENOTES THE COHERENCE MODELING TASK.

Methods	Prompt1	Prompt2	Prompt3	Prompt4	Prompt5	Prompt6	Prompt7	Prompt8	Average
EASE(SVR)*	0.781	0.621	0.630	0.749	0.782	0.771	0.727	0.534	0.699
EASE(BLRR)*	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705
HISK+BOSWE*	0.845	0.729	0.684	0.829	0.833	0.830	0.804	0.729	0.785
RNN	0.687	0.633	0.552	0.744	0.744	0.757	0.743	0.553	0.675
RL1	0.766	0.659	0.688	0.778	0.805	0.791	0.760	0.545	0.724
LSTM	0.780	0.697	0.683	0.787	0.795	0.767	0.758	0.651	0.740
CNN+LSTM	0.821	0.688	0.694	0.805	0.807	0.819	0.808	0.644	0.761
LSTM-CNN-att	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
SKIPFLOW	0.832	0.684	0.695	0.788	0.815	0.810	0.800	0.697	0.765
R^2BERT	0.817	0.719	0.698	0.845	0.841	0.847	0.839	0.744	0.794
BERT	0.815	0.720	0.730	0.814	0.820	0.824	0.833	0.730	0.786
BERT+CM	0.838	0.731	0.733	0.816	0.826	0.845	0.836	0.742	0.796
BERT+RM	0.831	0.728	0.741	0.838	0.834	0.853	0.829	0.733	0.798
BERT+RM+CM	0.842	0.733	0.746	0.842	0.836	0.857	0.842	0.747	0.806

D. Results

In this part, the performance of the baselines and our method on the ASAP dataset are analyzed in detail. Table II shows the QWK scores of different methods on each prompt.

In general, neural network-based methods have achieved better results than statistical-based methods. Even so, the statistical model HISK+BOSWE gain a better QWK score on Prompt 1, reaching 0.845. To some degree, it shows that when the handcrafted features can adequately represent the information in the original text, better results can be obtained. We notice that EASE performs better than RNN which also shows well-designed handcrafted features are more effective than simple neural networks. Among the methods based on neural networks, the performance of RNN and LSTM is not as good as R^2BERT. This is because that there are hundreds of words making it difficult to learn long-term dependencies. Meanwhile, we observe that CNN+LSTM, SKIPFLOW, and LSTM-CNN-att outperform LSTM models, which means that the ensemble model can make up for the shortage of simple neural networks. Additionally, BERT based model outperforms all other neural models on the average QWK score, which indicates the pre-trained language model does well in capturing deep semantic features.

Compared with other baseline models, the average QWK scores on eight prompts show that our model achieves the best results. Our model outperforms R^2BERT by 1.5% in the average QWK score. The result demonstrates that through the multi-task learning framework, our model can capture more coherence and relevance information for the final score evaluation. On Prompt 4 and 5, R^2BERT achieves higher results, indicating that combining complementary objectives via dynamic weights can effectively enhance the performance of the scoring system. In particular, BERT with auxiliary tasks outperforms R^2BERT on each prompt except Prompt 4 and 5, which shows the effectiveness of the auxiliary tasks and the success in improving the performance of BERT on downstream tasks. Overall, BERT+RM+CM gains a higher average QWK score compared with the aforementioned neural models as well as the latest statistical model HISK+BOSWE.

E. Discussion

To further verify the effectiveness of our proposed auxiliary tasks, we conduct ablation experiments with different settings. The relevant results are illustrated in Table II.

We can see that both the auxiliary tasks improve the automated essay scoring performance remarkably. Compared with the baseline BERT, employing the coherence modeling task (BERT+CM) yields a result of 0.796 in averaged QWK score, which brings a 1.3% improvement. Meanwhile, employing the relevance modeling task (BERT+RM) individually outperforms the baseline by 1.5%. The two tasks behave differently on different prompts. The RM task performs better on Prompt 3 to Prompt 6, while the CM task does on the others. The results may be due to differences in the genre and guidelines of the essay. For example, In Prompt 5, students were asked to describe the mood created by the author in the memoir and use the relevant information in the source material to support the answer. Therefore, for Prompt 5, a high-scoring essay is expected to contain specific information about the memoir, and the details of the memoir mentioned in the written essay are more important than the coherence of the sentence. For Prompts 7 and 8, the type of essay is narrative. The guidelines for these two prompts require human raters to give the highest score to essays that are coherent and engage the reader's attention through telling a story. Accordingly, the CM task shows better performance as it captures the sequence of semantic changes. When we integrate the two auxiliary tasks together (BERT+RM+CM), the performance further improves to 0.806 in the QWK score on average. It is obvious that these two auxiliary tasks have brought great benefits to the scoring task.

V. Conclusion

In this work, we introduce an approach based on two auxiliary tasks to assess the quality of texts. We integrate the auxiliary tasks into a multi-task learning framework to benefit the scoring task. To verify the effectiveness of our proposed method, we compare our model with several methods on the ASAP dataset. Experimental results show that our model outperforms previous methods. Our work also shows that auxiliary tasks can enhance the performance of the BERT model for downstream tasks. For future work, we plan to explore more dimensions for the automated text scoring task.

Acknowledgment

The authors acknowledge National Natural Science Foundation of China (Grant No: 62176029), the Key Research Program of Chongqing Science and Technology Bureau (cstc2020jscx-msxmX0149), and Graduate Research and Innovation Foundation of Chongqing, China (Grant No.CYS21061). This work is also supported by the National Natural Science Foundation of China under Grant 62102316, in part by the NWPU Development Strategy Research Fund Project Grant 2022FZY16. This work is also supported by National Natural Science Foundation of China under Grant No. 62002226 and Zhejiang Provincial Natural Science Foundation of China under Grant No.LHQ20F020001.

References

[1] J. G. Borade and L. D. Netak, "Automated grading of essays: A review," in *International Conference on Intelligent Human Computer Interaction*. Springer, 2020, pp. 238–249.

[2] P. Phandi, K. M. A. Chai, and H. T. Ng, "Flexible domain adaptation for automated essay scoring using correlated linear regression," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 431–439.

[3] K. Taghipour and H. T. Ng, "A neural approach to automated essay scoring," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 1882–1891.

[4] F. Dong, Y. Zhang, and J. Yang, "Attention-based recurrent convolutional neural network for automatic essay scoring," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 153–162.

[5] M. Chen and X. Li, "Relevance-based automated essay scoring via hierarchical recurrent model," in *2018 International Conference on Asian Language Processing (IALP)*. IEEE, 2018, pp. 378–383.

[6] H. Zhang and D. Litman, "Co-attention based neural network for source-dependent essay scoring," *arXiv preprint arXiv:1908.01993*, 2019.

[7] M. Mesgar and M. Strube, "A neural local coherence model for text quality assessment," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4328–4339.

[8] X. Li, M. Chen, J. Nie, Z. Liu, Z. Feng, and Y. Cai, "Coherence-based automated essay scoring using self-attention," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, 2018, pp. 386–397.

[9] M. Chodorow and J. Burstein, "Beyond essay length: evaluating e-rater's performance on tofel essays," *ETS Research Report Series*, vol. 2004, no. 1, pp. i–38, 2004.

[10] M. D. Shermis and J. C. Burstein, *Automated essay scoring: A cross-disciplinary perspective*. Routledge, 2003.

[11] M. Cozma, A. Butnaru, and R. T. Ionescu, "Automated essay scoring with string kernels and word embeddings," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 503–509.

[12] M. Dascalu, W. Westera, S. Ruseti, S. Trausan-Matu, and H. Kurvers, "Readerbench learns dutch: building a comprehensive automated essay scoring system for dutch language," in *International Conference on Artificial Intelligence in Education*. Springer, 2017, pp. 52–63.

[13] R. Yang, J. Cao, Z. Wen, Y. Wu, and X. He, "Enhancing automated essay scoring performance via cohesion measurement and combination of regression and ranking," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1560–1569.

[14] R. Caruana, "Learning many related tasks at the same time with backpropagation," in *Advances in neural information processing systems*, 1995, pp. 657–664.

[15] ——, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[16] S. Thrun and J. O'Sullivan, "Discovering structure in multiple learning tasks: The tc algorithm," in *ICML*, vol. 96, 1996, pp. 489–497.

[17] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[18] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8599–8603.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[20] J. Liu, Y. Xu, and Y. Zhu, "Automated essay scoring based on two-stage learning," *arXiv preprint arXiv:1901.07744*, 2019.

[21] F. Nadeem, H. Nguyen, Y. Liu, and M. Ostendorf, "Automated essay scoring with discourse-aware neural models," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2019, pp. 484–493.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[27] Y. Farag, H. Yannakoudakis, and T. Briscoe, "Neural automated essay scoring and coherence modeling for adversarially crafted input," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 263–271.

[28] Z. Lin, H. T. Ng, and M.-Y. Kan, "Automatically evaluating text coherence using discourse relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 997–1006.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[30] Y. Wang, Z. Wei, Y. Zhou, and X.-J. Huang, "Automatic essay scoring incorporating rating schema via reinforcement learning," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 791–797.

[31] Y. Tay, M. Phan, L. A. Tuan, and S. C. Hui, "Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

Exp-SoftLexicon Lattice Model Integrating Radical-Level Features for Chinese NER

Lijie Li[1], Shuangyang Hu[1], Junhao Chen[1], Ye Wang[1,*], Zuobin Xiong[2]
[1] College of Computer Science and Technology, Harbin Engineering University, Harbin, China
[2] Department of Computer Science, Georgia State University, Atlanta, USA
{lilijie, sunyonge, junhaochen, wangye2020}@hrbeu.edu.cn, zxiong2@student.gsu.edu

***Abstract*—The Lattice series model using potential words information has been proved to be effective in Chinese Named Entity Recognition (NER). The recently proposed Simplified Lattice not only brings new baseline results, but also improves the inference speed of Lattice models. However, the Simplified Lattice fails to fully explore the rich information contained in the radical-level features of the character sequences. Moreover, the performance of the Simplified Lattice decreases dramatically as the length of entity increases. In this paper, we propose the SLRL-NER model that integrates word, character, and radical-level information to alleviate the above problems. Specifically, text Convolutional Neural Network (CNN) is used to extract the radical-level features. The original SoftLexicon set is expanded to refine the relative position information of characters in the candidate words to cope with the challenge of increasing entity length. Experiments on three datasets show SLRL-NER outperforms the state-of-the-art comparison methods.**

Keywords- Chinese NER; radical-level features; SoftLexicon

I. INTRODUCTION

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that aims at identifying entities from plain text segments and tagging them with types, such as person, location, organization, etc. Many high level tasks, such as question answering [1], knowledge graph construction [2], and information retrieval [3] are all inseparable from NER.

Due to the implicit word boundary of Chinese sentences, Chinese NER is more difficult than English NER. The existing Chinese NER methods are mainly classified into word-based and character-based models. In the word-based model, a Chinese Word Segmentation (CWS) system is firstly used to segment the original input sequence. However, the performance of word-based model has been proved to be worse than character-based model because of the word segmentation errors from the CWS system [4]. In the character-based model, segmentation errors are naturally avoided. But character-based model does not exploit word-level information in input sequence at all. To integrate words information into character-based model, many works have attempted to use external gazetteers to extract word information [5-7]. Yue Zhang and Jie Yang [4] proposed the Lattice-LSTM model which encodes the matched lexical words information into character sequence with a gate mechanism and achieved a great success. Next, Ma et al. [8] proposed the Simplified Lattice model, where they constructed a SoftLexicon set for each word and used a word-weighting strategy to fuse word-level information. However, the Simplified Lattice model still faces two challenges.

Firstly, to the best of our knowledge, existing lattice models do not explore the radical-level information inherent in the Chinese characters. Radicals originate from pictograms [9], which are the minimum semantic units for Chinese characters. Different from the character representations obtained by the pretrained language model, radical-level features are context-independent [10] and relates only to the character itself. Thus, they have additional intrinsic information that is not present in the pretrained character embeddings. Dong et al. [11] have shown that the radical-level features can effectively improve the performance of Chinese NER. However, they ignored the word information in the sequence which has been demonstrated to be very important for Chinese NER by many works.

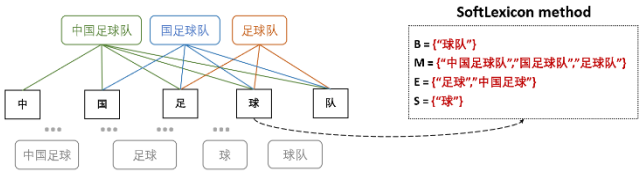


Fig. 1. The SoftLexicon method. "BMES" indicates the position of the character in the corresponding candidate word is: begin, middle, end and single.

Secondly, although the Simplified Lattice utilizes word information in a straightforward and efficient way, there is still a loss of position information for long words. As shown in Fig. 1, for the input sentence "中国足球队 (Chinese football team)", according to the position of the character in the word, the "middle" group word set of the character "球 (ball)" includes: "足球 (football team)", "国足球队 (Chinese football team)" and "中国足球队 (Chinese football team)". The character "球 (ball)" is the second, third and fourth character in three different words. However, the SoftLexicon method indistinguishably places the three different words of the character "球 (ball)" in the "Middle" group, which does not properly identify their specific relative positions within the words. As a result, this method loses a large amount of relative position information, which is proved to be essential for Chinese NER [12], and the problem becomes more serious as the length of entity grows.

DOI reference number: 10.18293/SEKE2022-037.
*Corresponding authors

In this paper, we consider all these issues systematically and present a novel exp-SoftLexicon Lattice Model Integrating Radical-Level Features for Chinese NER (SLRL-NER) to deal with these issues. **For radical-level information**: we elaborate a text CNN to extract radical-level information of characters from three different perspectives: radical, structural and unique position features. **To further exploit the position information of characters in candidate words**: we expand the original SoftLexicon module to enrich the relative position information of the "Middle" group to cope with the challenge of increased length of entity. Finally, we aggregate the radical-level and word-level information into the character representations to predict the named entity tags.

In summary, this paper makes the following contributions: (1) We propose a novel lattice structure to incorporate character-level, word-level and radical-level information of sentences for Chinese NER. Our model can capture both the inherent information and the rich context information for Chinese characters. (2) We introduce radical-level information into Chinese NER lattice model and design a text CNN module to exploit the radical-level features of characters from three different perspectives: radical, structural and unique position features. In addition, we apply an exp-SoftLexicon module to refine the relative position information of characters in the candidate words to deal with the challenge of increasing entity length. (3) Experimental results on three public Chinese NER datasets show that our model achieves better performance compared with the state-of-the-art methods.

II. BACKGROUND

In this section, we introduce some methods relevant to this paper, including radical-level features, character pre-trained methods and SoftLexicon module.

A. Radical-Level Features

Different from English, Chinese characters are pictographic, and most of them still retain the original pictogram meaning. In particular, the morphological information is mainly reflected in the radical, structural and unique position features of the characters [11]. Furthermore, the radical is the basic constituent unit of a Chinese character and contains both simplified and traditional forms, which is closely related to the meaning of the character. Structural features consist of the decomposition of a Chinese character and have the meta-information of the characters. The unique position contains the absolute position sequence of each character in writing order. For a monoradical character, we just use itself as its radical-level features. Fig. 2 shows the basic information and meaning of the Chinese character "烫 (hot)"[1]. The radical-level features of "烫 (hot)" include: (1) Radical feature: A simplified radical consisting of four strokes, which contains semantic information about the character; (2) Structural features: "烫 (hot)" is composed of three monoradical: "氵 (water)", "扬 (raise)" and "火 (fire)", which cover the meta-information of the character; (3) Unique position: The absolute position sequence of the unique writing order of the character "烫 (hot)", i.e. "汤 (soup)" and "火 (fire)".

Fig. 2. Basic information about character "烫 (hot)".

The fine-gained semantic information of "烫 (hot)" is extrated by exploring the radical-level features. We did not continue to excavate the Wubi features of characters because radical features are the smallest semantic units of Chinese characters while Wubi features usually do not have obvious semantic information [11].

B. Character Embedding

More and more works choose pre-trained model BERT [13] instead of word2vec to server as the character encoder. BERT fuses token embeddings, position embeddings and segment embeddings as input to obtain a better dynamic vector representation with a deep network and huge number of parameters. However, BERT only masks a single character in the Chinese sequence, which obviously loses part of the word-level semantic information. BERT-wwm-ext [14] is built on BERT by using a larger corpus to mask all the consecutive Chinese characters that composed the words, so that the embeddings of characters have the semantic information of the words.

C. SoftLexicon

For the input sentence $s = \{c_1, c_2, \dots, c_n\}$, an external gazetteer L is used to match the latent words corresponding to each character c_i. Then, a specific SoftLexicon [8] set is constructed for each character c_i: each word is assigned to the "BMES" word sets according to the position of the character c_i in the corresponding latent word. Then the word-level representation of the character can be obtained by integrating words information in the SoftLexicon set.

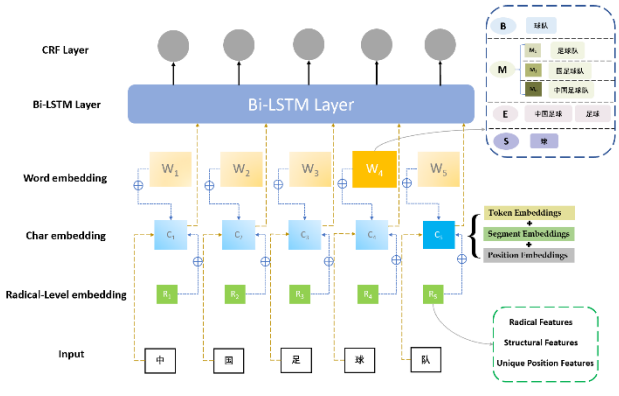


Fig. 3. The whole architecture of SLRL-NER. The top right part represents the exp-SoftLexicon set and ⊕ indicates the concatenation operation.

[1] from the online Xinhua dictionary at http://tool.httpcn.com/Zi/.

III. APPROACH

In this work, we propose the SLRL-NER model, which merges three types of information with different granularity: word, character and radical-level features to make full use of the semantic information in the input sequence and achieve excellent experimental results. The architecture of our model is shown in Fig. 3.

A. Radical-Level Representations Layer

The model input is a sentence $s = \{c_1, c_2, \dots, c_n\}$, and each character $c_i = \{r_1, r_2, \dots, r_m\} \in V_r$, where r_j denotes the radical-level features of the character. We use the radical-level lookup table V_r [11] which contains 4719 common Chinese characters. Each radical-level features r_j is represented by a dense vector $\boldsymbol{y}_j^r$:

$$\boldsymbol{y}_j^r = e^r(r_j), \tag{1}$$

where e_r denotes the radical-level features embedding lookup table. Then we can get the radical-level embedding matrix $\boldsymbol{O} = \{\boldsymbol{y}_1^r, \boldsymbol{y}_2^r, \dots, \boldsymbol{y}_m^r\}$ of the character c_i. To enable parallelization, the shape of radical-level embedding matrix $\boldsymbol{O}$ is set to be the form of $50 \times k$ where k is a hyper-parameter. For characters with more than k radical-level features, we perform a squeeze operation to take the top-k radical-level features, and for characters with less than k features, we proceed to random initialization to fill the feature matrix $\boldsymbol{O}$ to the fixed dimension. After obtaining the feature matrix $\boldsymbol{O} = \{\boldsymbol{y}_1^r, \boldsymbol{y}_2^r, \dots, \boldsymbol{y}_k^r\}$, the unsupervised CNN is employed to extract the radical-level features of the characters, as shown in Fig. 4.

Since the structural and unique position features of each character c_i mainly occur in pairs, the CNN apply filters $\boldsymbol{H} \in \mathbb{R}^{50\times 2}$ with a window size of 2. After x successive convolutions of $\boldsymbol{O}$, the radical-level features are extracted using maximum pooling and then represented as a 50-dimensional vector $\boldsymbol{y}^r$.

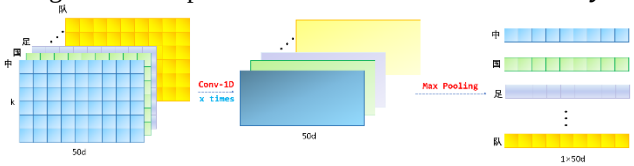


Fig. 4. Extracting radical-level features of characters in sequences using Convolutional Neural Networks.

B. Character Representation Layer

The character c_i in sentence is represented using a 768-dimensional dense vector $\boldsymbol{x}_i^c$:

$$\boldsymbol{x}_i^c = e^c(c_i). \tag{2}$$

Here, e^c denotes the character embedding lookup table, which is derived from BERT-wwm-ext [14]. BERT [13] masked the characters in the sequence without fully preserving the semantic information of the word. In BERT-wwm-ext, besides a character that constitutes a word, other parts that belong to the same word are masked, so that the pre-trained model is endowed with the semantic information of the word. As shown in Fig. 3, the final embedding representation of each character consists of token embeddings, position embeddings, and segment embeddings. After training with the full word masking strategy, the character embedding representation $\boldsymbol{x}^c$ is obtained.

C. Word Representation Layer

We build an exp-SoftLexicon set for each character in the sentence. Firstly, latent words are filtered by lexicon and classified into the six word sets "$BM_1M_2M_oES$" according to the position of the characters in the candidate words. The six word sets are constructed as follows:

$$\begin{aligned}
B(c_i) &= \{w_{i,k} \mid \forall w_{i,k} \in L, 1 \le i < k \le n\} \\
M_1(c_i) &= \{w_{j,k} \mid \forall w_{j,k} \in L, 2 \le j+1 = i < k \le n\} \\
M_2(c_i) &= \{w_{j,k} \mid \forall w_{j,k} \in L, 3 \le j+2 = i < k \le n\} \\
M_o(c_i) &= \{w_{j,k} \mid \forall w_{j,k} \in L, 3 \le j+2 < i < k \le n\} \\
E(c_i) &= \{w_{j,i} \mid \forall w_{j,i} \in L, 1 \le j < i \le n\} \\
S(c_i) &= \{c_i \mid \exists c_i \in L\}
\end{aligned} \tag{3}$$

Here, L denotes the lexicon we use in this work and $w_{i,j}$ denotes the matched word starting from the i-th character and ending at the j-th character. Additionally, if a word set is empty, a special word "NONE" is added to the empty word set. An example is shown in Fig. 3. By expanding the position of "Middle", our model distinguishes the candidate words in different positions in the "Middle" group more accurately, which reduces the loss of relative position of the SoftLexicon method and improves NER performance. After obtaining the expanded "$BM_1M_2M_oES$" word sets for each character, each word set is then compressed into a vector with fixed dimension. We obtain the representation of the word set W by a weighted strategy, then we concatenate all the six word sets representation to get the exp-SoftLexicon embedding z^w:

$$v^w(W) = \frac{6}{Z} \sum_{w \in W} z(w) e^w(w), \tag{4}$$

$$Z = \sum_{W \in B \cup M_1 \cup M_2 \cup M_o \cup E \cup S} \sum_{w \in W} z(w), \tag{5}$$

$$z^w = [v^w(B); v^w(M_1); v^w(M_2); v^w(M_o); v^w(E); v^w(S)], \tag{6}$$

where v^w denotes the weighting function, $e^w(w)$ denotes the embedding vector of word w, $z(w)$ denotes the number of occurrences of lexical word w in the statistical data, W denotes one of the "$BM_1M_2M_oES$" word sets corresponding to character c_i, Z is the sum of occurrence of all matched words in the six words sets, and z^w denotes the exp-SoftLexicon embedding vector corresponding to the character. In this work, the statistical dataset is made up of training and validation data. In addition, the frequency of w does not increase if w is covered by a subsequence of another matching lexical word. This avoids the problem that the frequency of a shorter word is always less than the frequency of the longer word containing it.

D. Bi-LSTM Layer

After obtaining three different granularity features of the input sequence, the next step is to preserve their individual

information as completely as possible and integrate them into the character representation. We finally choose to concatenate the representation vectors of the three, and the final representation of each character is obtained by:

$$\boldsymbol{x}^c \leftarrow [(\boldsymbol{x}^c; \boldsymbol{y}^r; \boldsymbol{z}^w)]. \tag{7}$$

Then, the final representations of characters are fed into Bi-LSTM. The definition of LSTM is as follows:

$$\begin{bmatrix} \boldsymbol{i}_t \\ \boldsymbol{f}_t \\ \boldsymbol{o}_t \\ \tilde{\boldsymbol{c}}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(W \begin{bmatrix} \boldsymbol{x}_t^c \\ \boldsymbol{h}_{t-1} \end{bmatrix} + b \right),$$

$$\boldsymbol{c}_t = \tilde{\boldsymbol{c}}_t \odot \boldsymbol{i}_t + \boldsymbol{c}_{t-1} \odot \boldsymbol{f}_t, \tag{8}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t).$$

where σ is the element-wise sigmoid function and $\odot$ denotes the product of elements, W and b are trainable parameters. Memory unit c can be considered as long-term memory and hidden state $\boldsymbol{h}$ as short-term memory. The backward LSTM shares the same definition as the forward LSTM, but models sequences in the opposite order. The hidden states of the i-th step of forward and backward LSTM are concatenated together to form the context-dependent representation of $\boldsymbol{c}_i$.

E. CRF Decoding Layer

Finally, to parse the dependencies between the continuous labels, we use a standard Conditional Random Field (CRF) [15] layer to perform sequence tagging. We use the following equation to calculate the score of the labels sequence: $y = \{y_1, y_2, \dots, y_n\}$:

$$Score(X, Y) = \sum_{i=1}^{n} P_{i,y_i} + \sum_{i=0}^{n} T_{y,y_{i+1}}, \tag{9}$$

where P is the output of the Bi-LSTM, which represents the emission score of the tag y_i in the sentence, and the transition matrix T denotes the transition probability from tag y_i to tag y_{i+1}.

In this work, we use the sentence-level negative log-likelihood loss function to train the model, and $L2$ regularization with the parameter λ is used to alleviate overfitting:

$$\mathcal{L} = Score(X, Y) - \log \sum_{y^i \in Y} e^s(x, y_i) + \frac{\lambda}{2} \|\theta\|^2. \tag{10}$$

IV. EXPERIMENT

A. Experimental Setup

Datasets. We evaluate the proposed model on three standard Chinese NER datasets, including **OntoNotes 4.0** [16], **Weibo** [17-18], and **Resume** [4]. **OntoNotes 4.0** is drawn from the news domain and contains four types of named entity. **Weibo** is built based on Chinese social media Sina Weibo, which contains PER, ORG, GEP, and LOC for both named entity and nominal mention. **Resume** is composed of resumes collected from Sina Finance. It is annotated with 8 types of named entities. We use the same dataset split and lexicon as the Simplified Lattice. The lexicon consists of 5.7k single-character words, 291.5k two-character words, 278.1k three-character words, and 129.1k other words. As for the pre-trained word embeddings, we also use the same one as the Simplified Lattice, which are pre-trained on Chinese Giga-Word using Word2vec [19].

Implementation Detail. In this work, the model is trained using stochastic gradient descent. The initial learning rate for Weibo is 0.005 and the other datasets are 0.0015. We apply dropout [20] to embedding layer with rate of 0.5 in order to avoid overfitting. Additionally, the hidden size of Bi-LSTM is set to 200 for small datasets **Weibo** and **Resume**, and 400 for larger dataset **OntoNotes 4.0**.

B. Experimental Results

Table 1 display the experimental results on three datasets. The character-based model gets performance boosted with the addition of softword and bichar features, which demonstrates it is critical to cooperate lexicon and character features in Chinese NER task. Another notable observation is that models use BERT encoder persistently outperform those without BERT. It indicates pre-trained character embeddings with context awareness are significant to this work. After replacing the encoder of Simplified Lattice with BERT-wwm-ext, we observe no significant change in performance, and even a decrease on the Resume dataset. This indicates that Simplified Lattice does not exploit the full potential of BERT-wwm-ext. GLYNN [21] improved the performance of Chinese NER by using a CNN encoder to integrate glyph features from character images, indicating that Chinese NER can benefit from the pictogram features. The SLRL-NER model integrates lexicon, character, and radical-level features leads to 0.47 and 0.25 increments of F1 score over the state of-the-art model on **OntoNotes 4.0** and **Resume**, respectively.

TABLE I MAIN RESULTS ON THE THREE DATASETS

Models	OntoNotes 4.0	Resume	Weibo		
	F1	*F1*	*NE*	*NM*	*F1*
Char-based	64.30	93.48	46.11	55.29	52.77
+bichar+softword	71.89	94.41	50.55	60.11	56.75
Peng and Dredze	–	–	55.28	62.97	58.99
He and Sun	–	–	54.50	62.17	58.23
SLK-NER	80.20	95.80	–	–	64.00
GLYNN+BERT	–	95.66	–	–	69.00
Lattice-LSTM	73.88	94.46	53.04	62.25	58.79
Simplified Lattice	75.64	95.53	59.08	62.22	61.42
Simplified Lattice+BERT	82.81	96.11	70.94	67.02	70.50
Simplified Lattice+BERT-wwm-ext	82.85	95.98	71.07	67.64	70.77
SLRL-NER	**83.28**	**96.36**	**72.15**	**68.62**	**71.90**

In the results on **Weibo**, NE, NM and Overall denote F1 scores for named entities, nominal entities (excluding named entities) and both, respectively. Weibo is made up of short and informal texts created by users, so it is difficult to identify. Therefore, baseline models adopt multitask learning with character embedding feature [22], semi-supervised learning, and cross-domain learning [23]. However, the architecture of the above models is complicated and may bring noise to the task, so these models only achieve limited improvement. In contrast, we use exp-SoftLexicon module to fuse the information of multiple candidate words to efficiently reduce word boundary conflicts. Additionally, we fully consider the radical-level, character-level and word-level semantic information in the sentence to alleviate the data sparsity problem. The experimental results show that our method is superior on social media domain compared with the state-of-the-art methods, which has 1.40 F1 score improvement.

C. Robustness Research

We perform experiments to verify the robustness of SLRL-NER. The results are shown in Fig. 5. The NER task becomes more challenging as the length of the named entity increases. The F1 scores on all three datasets suffers certain decrease. Specifically, the performance of both Simplified Lattice and Lattice-LSTM showed a large degree of fluctuation and degradation. SLRL-NER presents only a small decrease under different testing environments. Compared with the Simplified Lattice and Lattice-LSTM, our method is more robust. All the experiments are conducted on a single GPU with Quadro RTX 8000.

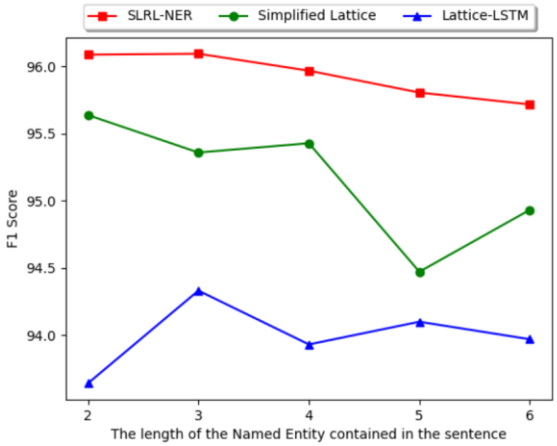


Fig. 5. Model performance against entity lengths. Batch size = 8 for SLRL-NER and Simplified Lattice models. Lattice-LSTM can only be trained with batch size = 1 due to its DAG structure.

D. Ablation Study

To investigate the contribution of each component of our method, we conducted ablation studies on all three datasets, the results are reported in Table 2. We find that: (1) Without radical-level features, F1 scores decrease 0.51, 1.36, and 0.46 on three datasets respectively. Radical-level features brings significant performance improvement. (2) After replacing the exp-SoftLexicon component with the standard SoftLexicon structure, the F1 score of SLRL-NER also decreases. It indicates the usefulness of finer grained relative position information in the input sequence.

TABLE II ABLATION STUDY OF SLRL-NER

Models	OntoNotes	Weibo	Resume
SLRL-NER	**83.28**	**71.90**	**96.36**
-radical-level features	82.77	70.54	95.90
- exp-Middle Group	82.89	71.16	96.21

V. CONCLUSION

In this work, we propose SLRL-NER, a noval lattice model which incorporates radical-level, character-level, and word-level information for Chinese NER. In order to leverage the radical-level features of the characters, we design a text CNN module to extract the radical-level information. The exp-SoftLexicon module is used to precisely capture the relative position information of characters in the potential words, which efficiently mitigates the challenge caused by the increase of entity length. Experiments on three Chinese NER datasets from different domains demonstrate our approach is superior compared with the state-of-the-art methods.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China under Grant No. 2020YFB1710200.

REFERENCES

[1] Dennis Diefenbach, Vanessa Lopez, Kamal Deep Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. Knowl. Inf. Syst., 55(3):529–569, 2018.

[2] LiYang LiuQiao, LiuYao DuanHong, et al. Knowledge graph construction techniques. Journal of computer research and development, 53(3):582, 2016.

[3] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In ACL, pages 167–176. The Association for Computer Linguistics, 2015.

[4] Yue Zhang and Jie Yang. Chinese NER using lattice LSTM. In ACL, pages 1554–1564. Association for Computational Linguistics, 2018.

[5] Wei Liu, Tongge Xu, QingHua Xu, Jiayu Song, and Yueran Zu. An encoding strategy based word-character LSTM for chinese NER. In NAACL-HLT, pages 2379–2389. Association for Computational Linguistics, 2019.

[6] Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. FLAT: chinese NER using flat-lattice transformer. In ACL, pages 6836–6842. Association for Computational Linguistics, 2020.

[7] Yuyang Nie, Yuanhe Tian, Xiang Wan, Yan Song, and Bo Dai. Named entity recognition for social media texts with semantic augmentation. In EMNLP, pages 1383–1391. Association for Computational Linguistics, 2020.

[8] Ruotian Ma, Minlong Peng, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. Simplify the usage of lexicon in chinese NER. In ACL, pages 5951–5960. Association for Computational Linguistics, 2020.

[9] Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. Radical embedding: Delving deeper to chinese radicals. In ACL, pages 594–598. The Association for Computer Linguistics, 2015.

[10] Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. Component-enhanced chinese character embeddings. In EMNLP, pages 829–834. The Association for Computational Linguistics, 2015.

[11] Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In NLPCC/ICCPOL, volume 10102 of Lecture Notes in Computer Science, pages 239–250. Springer, 2016

[12] Mengge Xue, Bowen Yu, Tingwen Liu, Yue Zhang, Erli Meng, and Bin Wang. Porous lattice transformer encoder for chinese NER. In COLING,

pages 3831–3841. International Committee on Computational Linguistics, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, pages 4171–4186. Association for Computational Linguistics, 2019.

[14] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese BERT.CoRR, abs/1906.08101, 2019.

[15] John D. Lafferty, Andrew McCallum, and FernandoC. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In ICML, pages 282–289. Morgan Kaufmann, 2001.

[16] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium, 2011.

[17] Nanyun Peng and Mark Dredze. Named entity recognition for chinese social media with jointly trained embeddings. In EMNLP, pages 548–554. The Association for Computational Linguistics, 2015.

[18] Hangfeng He and Xu Sun. F-score driven max margin neural network for named entity recognition in Chinese social media. In EACL (2), pages 713–718. Association for Computational Linguistics, 2017

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In ICLR (Workshop Poster), 2013.

[20] Nitish Srivastava, Geoffrey E. Hinton, AlexKrizhevsky, Ilya Sutskever, and Ruslan Salakhut-dinov. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res.,15(1):1929–1958, 2014.

[21] Chan Hee Song and Arijit Sehanobish. Using Chinese glyphs for named entity recognition (student abstract). In AAAI, pages 13921–13922. AAAI Press, 2020.

[22] Nanyun Peng and Mark Dredze. Improving named entity recognition for chinese social media with word segmentation representation learning. In ACL. The Association for Computer Linguistics, 2016.

[23] Hangfeng He and Xu Sun. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In AAAI, pages 3216–3222. AAAI Press, 2017.

AESPrompt: Self-supervised Constraints for Automated Essay Scoring with Prompt Tuning

Qiuyu Tao
Department of Computer Science
Chongqing University
Chongqing, China
TaoQiuyu@cqu.edu.cn

Jiang Zhong
Department of Computer Science
Chongqing University
Chongqing, China
zhongjiang@cqu.edu.cn

Rongzhen Li
Department of Computer Science
Chongqing University
Chongqing, China
lirongzhen@cqu.edu.cn

***Abstract*—Automated essay scoring(AES) aims to automatically assign scores to essays based on the quality of writing. Previous approaches have made many attempts with pre-trained BERT for essay scoring and achieved the state-of-the-art. However, these approaches mainly rely on the high computation cost and ignore the high similarity between text representations. In this paper, we propose a lightweight prompt-tuning framework, AESPrompt, to capture the significant semantic features of the text efficiently. We construct one continuous prompt for each layer of the frozen language model to help the language model understand the essay scoring task. Specially, we design task-related self-supervised constraints to capture discourse structure in terms of coherence and cohesion further to enhance the generalization and discourse awareness of the prompt. Experimental results on the public dataset ASAP illustrate that our approach performs competitively in the full data settings and outperforms in one-shot data settings significantly compared with fine-tuning BERT.**

***Index Terms*—Automated Essay Scoring, BERT, Prompt Tuning**

I. Introduction

Automated essay scoring(AES) aims to assign a score based on the essay quality, for essays written on a specific topic. AES is a necessary task in educational applications which can provide an efficient approach to score large-scale text and reduce human efforts remarkably. Early works in AES mainly leveraged the handcraft features such as such as grammaticality, spelling errors, and the length of essays [1]. Although AES systems based on feature engineering are explainable, it is expensive to design scoring rubrics for the new writing topics.

Existing works are mainly based on Convolution Neural Network(CNN) and Recurrent Neural Network(RNN) to learn text representations. The key challenge of neural-network-based AES systems is to learn a better text representation that can capture deep semantic features as much as possible. However, those neural networks requires more annotated essays for training. Shallow neural networks trained on limited samples show poor performance to capture deep semantics of texts which may obstruct an AES system to further ensure correct scoring.

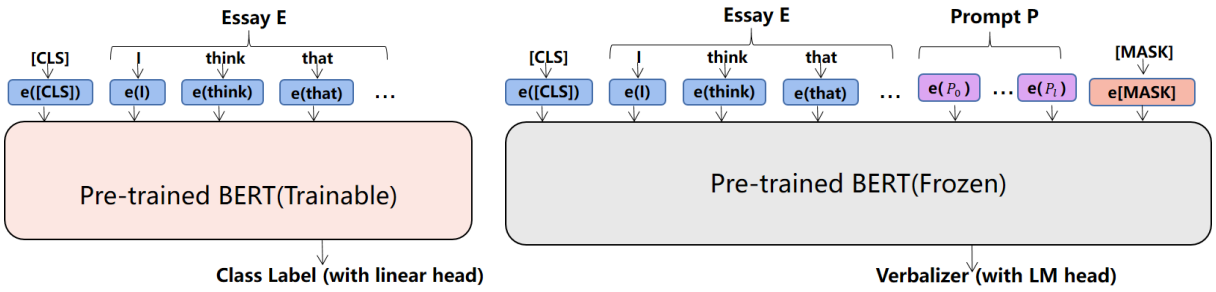
DOI reference number: 10.18293/SEKE2022-095

In recent years, pre-trained language models(PLMs) such as the BERT [2], have improved performance in many natural language downstream tasks such as text classification and sentiment analysis, which shows its extraordinary representation ability. The key component of the BERT model is the self-attention mechanism [3], which can capture the relationship between any words in the essay even the long text. Although some prior approaches utilize methods to fine-tune BERT [4], [5], these approaches are dependent on high computation costs which tune all model parameters and need to store a full copy of the model for each writing topic. Besides, previous works ignore the significant gap between pre-training and downstream tasks, which restricts BERT from reaching its full potential.

The prompt-based tuning method is proposed to narrow the gap between downstream and pre-train tasks [6]. Unlike the traditional fine-tuning method, prompt-based tuning reformulates natural language understanding (NLU) as a masked language modeling task, as the fig 1(b) shows. Formally, we make a template function $x_{prompt} = \tau(x)$ to concatenate the input with prompts and one answer slot [Z]. For a masked language model, the slot [Z] is fill with the [MASK] token. For instance, when it applies to AES, we can simply define a template $\tau(x) =$ "$[x]$ $Assign\ the\ essay\ on\ a\ scale\ of$ 0 to $4.[MASK]$". By feeding a supervised example $\{x_{prompt}, y\}$ into the masked language model M, we can determine the essay score with PLMs predicting '1' or '2' at the mask position. Prompt-based tuning can help PLMs better understand the task, meanwhile introducing no new parameters within PLMs and making it easier to fine tune.

PLMs with the fine-tuning need to store all model parameters for each downstream task. However, discrete prompts can be sub-optimal for the continuous PLMs. A recent line of work proposes the prompt-tuning paradigm [6]–[8] to adapt large PLMs to downstream tasks cheaply. Prompt tuning freezes all the parameters in PLMs but only tunes the prompts, making the method more efficient. Also, the prompts are initialized randomly and learned end-to-end, reducing the cost of manually designing the template.

Motivated by the above observations, in this paper, we propose AESPrompt, a novel prompt-tuning framework for



(a) Fine Tuning (b) Prompting

Fig. 1: Paradigms of fine-tuning(figure a) and Prompting(figure b) for automated essay scoring.

AES. We first construct a multi-layer prompt that prepends a continuous embedding into the input sequence for each layer of PLMs. Specifically, the prompts in different layers are independent, bringing more tunable parameters than other prompt-tuning methods. At the same time, it is still much smaller than the full PLMs. To further inject essay scoring self-supervised constraints into the prompt, we propose AES-related self-supervised learning to constrain the prompt including the "Discourse Indicator shuffle" and the "Paragraph Reordering Detection". Our main contributions can be summarized as follows:

- We propose AESPrompt, a prompt-tuning framework for the essay scoring task. To the best of our knowledge, this is the first approach to incorporate a prompt-based method for essay scoring.
- To better optimize the continuous prompts, we propose AES-related self-supervised constraints, including discourse indicator and paragraph order.
- We conduct experiments on the ASAP dataset with the $BERT_{base}$ model. Experimental results not only illustrate the effectiveness of AESPrompt in full data settings but also reinforce the stability in low-resource settings.

II. Related Work

A. Automated Essay Scoring

Early studies about the AES task starts with feature engineering. The systems use textual features designed by human experts [1]. The latter type of researches use deep neural networks to extract features automatically. Taghipour and Ng [9] first propose a neural method based on CNN and LSTM to learn essay representation for essay scoring. Many works improve AES based on that [10]–[14] BERT has achieved state-of-art results on many downstream NLP tasks. some prior works find BERT sentence embedding is useful for the ASAP data [5], [13], [15], [16]. TSLF [15] calculates the semantic score, coherence score, and prompt-relevant score during the first stage, and then concatenates handcraft features for further training. While Nadeem et al. [13] finds that token and sentence embedding from BERT makes no significant improvement. Their work explores discourse-based pre-training tasks and contextualized embedding and proposes a discourse-aware neural framework. R^2BERT [5] model is proposed to solve the essay scoring task and essay ranking task jointly. The model is fine-tuned by a multi-loss approach which combines the scoring MSE loss and a ranking error loss based on ListNet.

B. Prompt-based learning

Prompt-based methods are inspired by the birth of GPT-3 [17], which reformulate downstream tasks to language modeling tasks with textual templates and a verbalizer. The prompting method is first applied as a knowledge probe [18]. However, handcraft prompts heavily depend on the experience of designers, so some works explore automatically generating discrete prompts via gradient-based search [7], [19]. Shin et al. [8]propose an approach to generate prompts in vocabulary automatically. Compared with the fine-tuning method, the prompting method freezes all model parameters, which may lead to the volatile performance of the model in many cases. [20], [21]. Prompt tuning is proposed to only tune the continuous prompts and outperforms prompting in many tasks. Han et al. [22] propose prompt tuning with rules for text classification, Chen et al. [23] applied prompt-tuning with synergistic optimization on relation extraction. Recently, some works focus on optimizing continuous prompts for every layer of pre-trained model [24], [25]. In this paper, we propose a novel Prompt Tuning framework for the AES task. Besides, we inject AES-related self-supervised to constraint the prompt. To our knowledge, we are the first to apply prompt-based method to Automated essay scoring.

III. AESPrompt

In this section, we introduce our AESPrompt framework as shown in Fig 2.

A. Prompt Encoder

The overall framework involves a language model to learn text representation, which is then used for essay scoring. Following the deep prompt tuning approach as in P-Tuning-v2 (PT2) [26] which is an NLU version of prefix-tuning [24]. PT2 keeps all pre-train language model parameters frozen and only tunes the prompt parameters. We regard the AES task as a regression task and predict the score via [CLS] token. First,

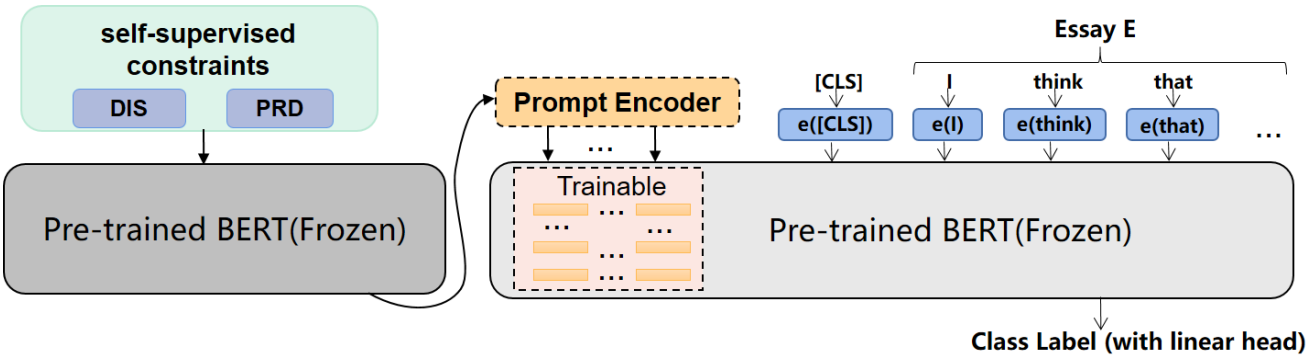


Fig. 2: Model architecture of AESPrompt. We design two self-supervised constraints for Prompt tuning including **D**iscourse **I**ndicator **S**huffle(DIS) and **P**aragraph **R**eordering **D**etection(PRD).

for a given sample essay $x = \{w_1, w_2, \ldots, w_n\}$, it is should be tokenized into a new sequence $\tilde{x} = \{[CLS], e_1, e_2, \ldots, e_n\}$, where n is the number of words and [CLS] is a special classification token. We conduct M to obtain the hidden representations of the inputs $h = M(\tilde{x}) \in \mathbb{R}^{|\tilde{x}| \times d}$, where $|\tilde{x}|$ is the sequence length. To reduce the objective gap between pre-training and AES, PT2 prepends prompt for each layer of M as additional keys $K^P \in \mathbb{R}^{L \times d}$ and values $V^P \in \mathbb{R}^{L \times d}$ to the multi-head self-attention mechanism, where L is the prompt length and d is the dimension of word embedding. The new text representation is obtained through attention as the show below:

$$\mathrm{Att}(Q, K) = \mathrm{softmax}(\frac{Q[K^P, K]^T}{\sqrt{d}}) \quad (1)$$

$$V_{att}(Q, K, V) = \mathrm{Att}(Q, K) \cdot [V^P, V] \quad (2)$$

where $[,]$ refers to the concatenation operation. Then, the hidden representation is mapped to CLS token $h_{[CLS]}$. Since score ranges are different from each other, during training the gold scores are normalized into the range of [0,1] first. Then during the test process, map the predicted scores to the original score ranges. We can thus conduct a linear layer with activation function to project the $h_{[CLS]}$ to a scalar value as formula (3), where W is weight matrix and b is a bias initialized by the mean gold score of training data [9].

$$\acute{y} = \mathrm{Sigmoid}\left(\mathrm{W}h_{[\mathrm{cls}]} + b\right) \quad (3)$$

B. Prompt Tuning with self-supervised constraints

Simply random initializing prompts with continuous embeddings brings difficulties to optimization. Fortunately, neural networks can utilize related tasks to improve performance through pre-training. To inject essay scoring self-supervised constraints into prompt, we design self-supervised learning with discourse indicator shuffle and paragraph reordering to pre-train our prompts. The self-supervised constraints further enhance the prompt's generalization and document structure awareness. Instead of using additional data, we generate pre-train data from the original data as shown in Fig 3. We introduce the details in the following sections.

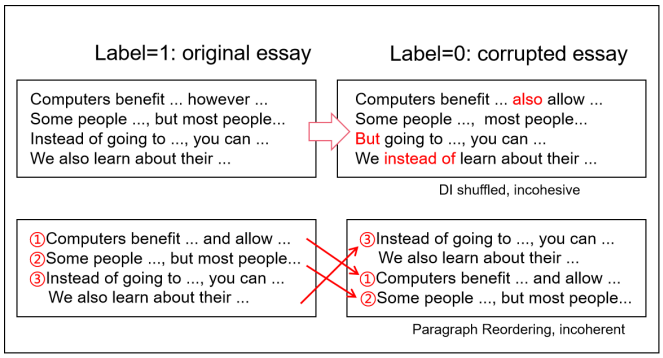


Fig. 3: Proposed self-supervised constraints which utilizing coherent/cohesive and incoherent/incohesive texts for Prompt Tuning

1) Discourse Indicator Shuffle: To strengthen the bidirectional representation on the AES task, we construct the discourse indicator shuffle task. The discourse indicator refers to the conjunction indicating the relationship between sentences (e.g. "however", "else" and "while"). Though DIs has a somewhat empty meaning, without sufficient DIs in a piece of writing, a text would lack logic and the connection between different sentences and paragraphs will be unfluent. We design a binary classification to detect whether discourse indicators are shuffled or not. To simplify, for one DI token is chosen, 1) we replace the token with other DI randomly 60% with the time. 2) delete the DI 20% with the time, 3) unchanged the token 20% of the time. For example, "they use an online catalog because it's cheaper" is cohesive. "they use an online catalog but it's cheaper" and "they use an online catalog, it's cheaper" is incohesive.

2) Paragraph Reordering Detection: The AES task is based on understanding not only the relationship between two sentences but also paragraphs while the relationship between paragraphs is not modeled directly when pre-training. With the hypothesis that many student essays follow a logical structure like, "introduction-body-conclusion". We propose to reorder paragraphs that divide the document into three parts and each part consists of one or more complete sentences.

And then, we permute them into a certain permutation. Since the permutations show great influence in representation learning, we choose the permutations with the maximal average Hamming distance [27]. We use three possible permutations $P = \{(1,2,3),(2,3,1),(3,1,2)\}$ in our experiments. We label the essay with permutation $P_i = \{(1,2,3)\}$ as coherent and label another two permutations as incoherent.

C. Training

a) The scoring Task: is treated as a regression task. We use the Adam optimization algorithm to minimize the mean squad error (MSE) function. Given a training essays of size N, y_i and $\acute{y_i}$ are the corresponding gold and predicted score for i-th eassy, separately. The loss function is shown in Formula(4) :

$$\mathrm{L_s(x_i, y_i)} = \mathrm{MSE(y_i, \acute{y_i})} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \acute{y_i})^2 \tag{4}$$

b) Self-supervised Constraints: We create the training instances for self-supervised constraints as section 3.2 mentioned. Since the tasks are treated as binary classification tasks, we use the cross-entropy loss function for self-supervised constraints, respectively. Where $\mathrm{c_i}$ and $\acute{c_i}$ are the corresponding gold and predicted label of the input essay $\tilde{x_i}$. Note that $\mathrm{c_i}$ is automatically assigned in the corruption process where an original essay has a label of 1 and an artificially corrupted essay has a label of 0.

$$\mathrm{L_d}(\tilde{x_i}) = -\sum_{j=1}^{M} \mathrm{c_i}\log(\acute{c_i}) - (1-\mathrm{c_i})\log(1-\acute{c_i}) \tag{5}$$

IV. Experiments

TABLE I: Details of ASAP Dataset.

Set	Score Range	Type of essay	Mean length
1	2-12	persuasive	350
2	1-6	persuasive	350
3	0-3	source dependent response	150
4	0-3	source dependent responses	150
5	0-4	source dependent responses	150
6	0-4	source dependent responses	150
7	0-30	narrative	250
8	0-60	narrative	650

In this section, we first introduce the ASAP dataset and evaluate metrics. And then we describe the experimental setup and present the results.

A. Dataset and Metrics

The Automated Student Assessment Prize(ASAP) dataset is provided by a Kaggle competition which contains eight different essay sets written by students from grade 7 to grade 10. This dataset has become the most widely used in the field of AES which is composed of 12976 labeled essays. More details about the ASAP are summarized in Table I.

https://www.kaggle.com/c/asap-aes/data

We employ the quadratic weighted kappa(QWK) as the evaluation metric, which is the official evaluation metric adopted by ASAP competition. Quadratic weight kappa measures the agreement between gold scores and automated scores. The QWK is calculated as follows, an N by N weight matrix is calculated first according to formula (6):

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{6}$$

where i refers to the gold score, j refers to the predicted score (assigned by the AES model) and N is the total number of essays. Second, we construct the confusion matrix O, that $O_{i,j}$ corresponds to the number of essays rated i by human rater and rated j by AES model. Then, an expected matrix E is calculated as the outer product between gold scores and predict scores. The matrix E is normalized such that E and O have the same sum. Finally, from the three matrices, the QWK is calculated as formula (7):

$$k = 1 - \frac{\sum_{i,j} W_{i,j}O_{i,j}}{\sum_{i,j} W_{i,j}E_{i,j}} \tag{7}$$

B. Experimental settings

We explore the following setups to train AESPrompt models for ASAP essays :

1) Training using only ASAP essay data;
2) Pretraining with either DIS or PRD data, followed by training with the essay data.
3) Pretraining with DIS and PRD data, followed by training with essay data.

For all our experiments, we use the "BERT-base-uncased" model as the base model, and the training is implemented on PyTorch with an Nvidia A6000 GPU. In our work, the linear learning rate policy is used to tune the parameters, and the max learning rate is set to 1e-3. In full data experiments, closely following the settings as [9], we conduct five-fold cross-validation with a 3:1:1 split for training, validation, and test to evaluate our method. We report the average QWK across the five folds. For one-shot data experiments, we follow Gao et. al [7], which assumes development data has the same size as train data to select model and hyper-parameters. And we repeat the sampling of one-shot labled data 5 times and the average results are reported. Consistently, we set prompt length to 40, as a result, the tunable parameters are only 80k, compared with 110M parameters of BERT fine-tuning, our method only needs to store additional 0.7% for each essay set.

C. Main Results

We evaluate the performance on the ASAP dataset, the main results are shown in Table II. To put our results in perspective, we compare our method with several baseline models. Enhanced AI Scoring Engine (EASE) is a statistical model based

https://github.com/huggingface/transformers
https://github.com/edx/ease

TABLE II: The performance (QWK) of all comparison methods on ASAP dataset. The best measures are in bold. * denotes statistical model.

settings	Models	1	2	3	4	5	6	7	8	avg
full-data	EASE(SVR)*	0.781	0.621	0.630	0.749	0.782	0.771	0.727	0.534	0.699
	EASE(BLRR)*	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705
	CNN+LSTM	0.821	0.688	0.694	0.805	0.807	**0.819**	0.808	0.644	0.761
	LSTM-CNN-att	0.822	0.682	0.672	**0.814**	0.803	0.811	0.801	0.705	0.764
	SKIPFLOW	**0.832**	0.684	**0.695**	0.788	**0.815**	0.810	0.800	0.697	0.765
	BERT	0.809	0.661	0.692	0.808	0.800	0.801	**0.834**	0.720	0.765
	P-Tuning-V2	0.781	0.640	0.677	0.758	0.794	0.798	0.825	0.717	0.749
	AESPrompt(DIS)	0.802	0.680	0.680	0.765	0.807	0.801	0.825	0.722	0.760
	AESPrompt(PRD)	0.788	0.650	0.670	0.784	0.793	0.803	0.826	**0.727**	0.755
	AESPrompt(ALL)	0.808	**0.689**	0.685	0.790	0.803	0.806	0.833	0.724	**0.767**
one-shot	BERT	0.625	0.545	0.431	0.515	0.647	0.485	0.664	0.646	0.572
	P-Tuning-V2	0.568	0.522	0.554	0.649	0.681	0.610	0.664	0.613	0.607
	AESPrompt(DIS)	0.680	0.532	0.585	0.660	0.698	0.617	0.669	0.598	0.630
	AESPrompt(PRD)	0.658	0.542	0.566	0.667	0.685	0.613	0.678	0.603	0.627
	AESPrompt(ALL)	**0.682**	**0.544**	**0.590**	**0.672**	**0.701**	**0.622**	**0.683**	**0.620**	**0.639**

on hand-crafted features followed by support vector regression (SVR) and bayesian linear ridge regression (BLRR) [28]. CNN+LSTM [9] is proposed to assemble CNN and LSTM to predict the essay rating. CNN-LSTM-Att [12] introduces hierarchical neural networks with attention mechanism to learn the representation of essays. SKIPFLOW [14] considers the coherence when learning text representations. BERT [2] is employed as an encoder for the AES task. P-Tuning-v2 [26] performs deep prompt tuning, which prepends prefix prompts in the input of model's hidden layer. AESPrompt(DIS) employs the discourse indicators shuffle constraint. AESPrompt(PRD) only includes the paragraph reordering detection constraint. AESPrompt(ALL) employs both two constraints. The BERT fine-tuning and SKIPFLOW give a strong baseline, the average QWK across eight sets is 0.765 in the full-data setting. LSTM-CNN-att and SKIPFLOW both are hierarchical models which explicitly capture the adjacent semantics in each essay. So they perform better in set 1, 3, 4, 5 and 6. We can see that the AESPrompt method slightly outperforms in a resource-rich setting. AESPrompt shows obvious advantages on two narrative essay sets(set 7 and 8). By incorporating the self-supervised constraints, the proposed framework dramatically improves the accuracy of PT2 at an average of 0.018 QWK.

To further evaluate the potential of our method, we conduct one-shot setting experiments on the ASAP dataset. We compare our approach with BERT fine-tuning and P-Tuning-v2. AESPrompt significantly outperforms the BERT fine-tuning and P-tuning-v2 in one-shot settings, which shows AESPrompt appears to be more beneficial in low-resource settings. Specifically, AESPrompt can obtain grains of up to 11.7% improvement on average compared with BERT fine-tuning. We can find out that AESPrompt outperforms in all sets. What is more, We also observe that our results suffer from high variance. The performance fluctuates up to 15% QWK under different randomly sampled D_{train} and D_{dev}. In one-shot settings, truncating text that exceeds the length may have a great impact on AESPrompt. We will explore these problems in the future.

TABLE III: Comparison of Runtime and Memory. TR means the total training time on the train set and IPS means inference runtime per each test sample. Parameters refer to the number of tuned parameters.

Model	TR	IPS	Parameters
BERT fine-tuning	256	0.067	110M
P-Tuning-v2	179	0.062	80k
AESPrompt	185	0.062	80k

D. Ablation Study

We explore the effects of the self-supervised constraints for the AESPrompt, by removing each of them individually. These self-supervised constraints include: discourse indicators shuffle, and Paragraph reordering detection. As shown in Table II, after removing one of them from AESPrompt, the performance decrease a lot. These indicate that the self-supervised constraints we proposed can enhance the prompts discourse awareness from paragraph level and discourse indicator level. In addition, the performance of AESPrompt(PRD) is worse than AESPrompt(DIS) which indicates that using the RPD constraint alone may fail to benefit the general regression model.

E. Runtime and Memory

Our secondary evaluation is based on the runtime and resource usage which means the total number of parameters. In summary, we main compare BERT fine-tuning, P-Tuning-v2 and AESPrompt model as Table III shows. Firstly, we estimate the total tuned parameters for the three models. Then, We take essay set 1 as an example to compare the model runtime. Since the prompt tuning needs more training epochs to converge than BERT fine-tuning that we record the total training time for each method. And we record the inference time on one sample to compare the efficiency of inference. In our approach, we freeze all parameters in the language model that reduce the storage and computation consumption. It's practical in real educational scenarios that AESPrompt can reach a reasonable

performance on scoring task only needs to store additional 80k parameters for a new essay scoring set.

V. Conclusion

In this work, we propose a lightweight prompt tuning framework with self-supervised constraints, AESPrompt, for automated essay scoring. Specifically, we propose two AES-related self-supervised constraints to pre-train the prompt which further reduces the intrinsic gap between the language model distribution and the target data distribution. In this way, both full-data and the one-shot performance can be boosted. Compared with standard BERT fine-tuning, our method is lightweight, which only tunes 80k parameters compared with 110M. Experimental results show that the proposed method achieves significant improvement on one-shot AES and competitive results on full-data AES. In this case, our approach is meaningful for the practical of PLMs in automated essay scoring. In the future, we plan to explore how to design unified task formats and the corresponding auxiliary task on eight sets.

Acknowledgment

The authors acknowledge National Natural Science Foundation of China (Grant No: 62176029), the Key Research Program of Chongqing Science and Technology Bureau (cstc2020jscx-msxmX0149), and Graduate Research and Innovation Foundation of Chongqing, China (Grant No.CYS21061). This work is also supported by the National Natural Science Foundation of China under Grant 62102316, in part by the NWPU Development Strategy Research Fund Project Grant 2022FZY16.

References

[1] H. Yannakoudakis, T. Briscoe, and B. Medlock, "A new dataset and method for automatically grading esol texts," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 180–189.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, 2017.

[4] E. Mayfield and A. W. Black, "Should You Fine-Tune BERT for Automated Essay Scoring?" in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Seattle, WA, USA → Online: Association for Computational Linguistics, 2020, pp. 151–162.

[5] R. Yang, J. Cao, Z. Wen, Y. Wu, and X. He, "Enhancing Automated Essay Scoring Performance via Fine-tuning Pre-trained Language Models with Combination of Regression and Ranking," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, 2020, pp. 1560–1569.

[6] T. Schick and H. Schütze, "Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference," *arXiv preprint arXiv:2001.07676*, 2021.

[7] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *ACL/IJCNLP (1)*, 2021.

[8] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts," *arXiv preprint arXiv:2010.15980*, 2020.

[9] K. Taghipour and H. T. Ng, "A Neural Approach to Automated Essay Scoring," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, pp. 1882–1891.

[10] D. Alikaniotis, H. Yannakoudakis, and M. Rei, "Automatic Text Scoring Using Neural Networks," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 715–725, 2016.

[11] M. Cozma, A. Butnaru, and R. T. Ionescu, "Automated essay scoring with string kernels and word embeddings," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 503–509.

[12] F. Dong, Y. Zhang, and J. Yang, "Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 153–162.

[13] F. Nadeem, H. Nguyen, Y. Liu, and M. Ostendorf, "Automated Essay Scoring with Discourse-Aware Neural Models," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 484–493.

[14] Y. Tay, M. Phan, L. A. Tuan, and S. C. Hui, "Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[15] J. Liu, Y. Xu, and Y. Zhu, "Automated Essay Scoring based on Two-Stage Learning," *arXiv preprint arXiv:1901.07744*, 2019.

[16] A. Sharma, A. Kabra, and R. Kapoor, "Feature enhanced capsule networks for robust automatic essay scoring," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 365–380.

[17] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[18] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language Models as Knowledge Bases?" *arXiv preprint arXiv:1909.01066*, 2019.

[19] K. Hambardzumyan, H. Khachatrian, and J. May, "Warp: Word-level adversarial reprogramming," *arXiv preprint arXiv:2101.00121*, 2021.

[20] B. Lester, R. Al-Rfou, and N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 3045–3059.

[21] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[22] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "PTR: Prompt Tuning with Rules for Text Classification," *arXiv:2105.11259*, 2021.

[23] X. Chen, N. Zhang, X. Xie, S. Deng, Y. Yao, C. Tan, F. Huang, L. Si, and H. Chen, "Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction," *arXiv preprint arXiv:2104.07650*, 2021.

[24] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597.

[25] G. Qin and J. Eisner, "Learning How to Ask: Querying LMs with Mixtures of Soft Prompts," *arXiv preprint arXiv:2104.06599*, 2021.

[26] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[27] Y. Cao, H. Jin, X. Wan, and Z. Yu, "Domain-adaptive neural automated essay scoring," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1011–1020.

[28] P. Phandi, K. M. A. Chai, and H. T. Ng, "Flexible domain adaptation for automated essay scoring using correlated linear regression," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 431–439.

Increasing Representative Ability for Topic Representation

Rong Yan, Ailing Tang, Ziyi Zhang
College of Computer Science, Inner Mongolia University
Inner Mongolia Key Laboratory of Mongolian Information Processing Technology
National & Local Joint Engineering Research Center of Intelligent Information Processing Technology for Mongolian
Hohhot 010021, China
Email: csyanr@imu.edu.cn

***Abstract*—As for standard topic model, such as LDA (Latent Dirichlet Allocation), each topic is generally depicted by a weighted word set, where the high-ranked words are deemed more representative. Meanwhile, the probability of each word is considered as the ability to represent the semantic contribution for the topic. However, few efforts are focused on enhancing the representative ability of the topic to support fine grained topic representation. In this paper, we propose a Word Topic Ware (WTW) model to take word inherent diversity characteristic into consideration, in order to screen out and enhance the more representative words for topic representation. Experimental results on three large datasets show that our proposed method can increase the representative ability for topic representation. In addition, our work will positively affect improving the quality of topic content analysis.**

***Index Terms*—Topic analysis, Text representation, Topic model, Latent Dirichlet Allocation.**

I. Introduction

Probabilistic topic model (PTM) family [1] offers a promising solution to discover and extract a mixture of latent topic set that occur in a large document collection. Under the bag-of-word assumption, topic modeling approaches, such as Latent Dirichlet Allocation (LDA) [1], are implicitly capture the document-level word co-occurrence patterns to reveal each latent topic as a multinomial distribution over a weighted word set through the statistical techniques [2]. Meanwhile, a weighted topic set that supposed to be semantic or representative is used to summarize and organize the semantic and hidden structures of documents. Thus, each document is forced to represent as the same specific combination of a topic set. The probability value of the specific word in each topic reflects its representative ability that we called semantic contribution degree. Generally, the bigger probability value of the word represents its bigger semantic capacity ability. Ideally, topics discovered by standard PTMs should be independent from each other under the assumption that the topic proportions are randomly drawn from a Dirichlet distribution. In this paper, we call it 'topic independence'. Therefore, the explanation of each topic should be single-minded and no ambiguity, that is to say, the topic representation results should maintain the topic reliability. Unfortunately, it is commonly seen that the same word often appears in different topics simultaneously in the real dataset. It makes this topic depiction manner unapparent. Meanwhile, it is very difficult to keep 'topic independence'. But the truth is that the standard PTM is really difficult to improve this scene. In summary, the reason lies in two aspects, including word frequency and word inherent diversity characteristic.

The word with high-frequency in the document collection is bound to appearing in topic-word distribution with higher rankings for most of the topics due to the 'bag-of-word' assumption. However, the semantic contribution degree of this kind of words are not in accord with its representative ability, but weaken topic independence. At the same time, they are general and popular in topic description, which are called the common words. However, as it will be seen later, a topic distribution under which a large number of words with higher probability is not always likely to be insignificant. Though most of these words are filtered out in standard PTMs, which are considered as common stop-words, there are still majority of common words undertake the supported role for the topic description, and it should be reserved for avoiding the semantic loss [3]. And at this point, it is inseparable from the word inherent diversity characteristic. In standard PTMs, such as LDA, the *top*-ranked word set of the specific topic is generally used to represent the topic description. Thus, the word inherent diversity characteristic lies in the semantic representative ability for the specific topic. As we all known, the outward manifestation of word inherent diversity characteristic lies in the probability of the word in each topic. To that end, intensive efforts have been invested on finding the appropriate method to discriminate the word inherent diversity characteristic [4].

However, few previous studies consider this discrimination from the real status of the same word in different topics, even though it is exactly the common word. To overcome the limitations of previously proposed methods, specifically, we propose a Word Topic Ware (WTW) model for taking account of word inherent diversity characteristic, in order to identify and refine the quantification of the meaningful representative words for topic representation. The main idea comes from the answers of the following two questions extensive: (1) Is the same word that in different topics represent the same semantic

DOI reference number:10.18293/SEKE2022-052

meaning? ($Q1$) (2) Is it correct that each latent topic with single semantic? ($Q2$)

The contribution of this paper is as follows. This paper proposed a WTW model to take word inherent diversity characteristic into consideration, in order to screen out and enhance the more representative words for topic representation. To the best of our knowledge, this is the first time considering the word inherent diversity characteristic for topic analysis.

The remainder of this paper is organized as follows. Section 2 reviews the related work on topic quality analysis. Section 3 analyses the incoherence and bias in PTMs. Section 4 presents our proposed method (WTW). Section 5 describes our experiments. In Section 6, we make a conclusion.

II. Related Work

In this paper we mainly discuss the topic quality problem of PTM, so we review the related research effort in this section. The current automated evaluations of PTM topical quality research mainly focus on two aspects: topic groups quality and individual topic quality.

Much effort has been devoted to automated evaluation approach for topic groups quality, and the mainly metrics include perplexity and topic coherence. The perplexity value lies to estimate the generalization capability of the model fit [5], and the lower value represents a higher performance of the model. However, this metric pays less attention to the semantic interpretability of the words composed a specific topic [6]. The real fact is that the lower perplexity of topics is not necessarily correlated to better coherence of topics, even negatively with topic interpretability [6]. Thus, topic coherence is considered as a supplement metric to evaluate the topic groups quality emphasised on understandability and interpretability [7]–[11]. Topic coherence can be estimated by the semantic similarity of topic words [7], [10], [11] or topic documents [11].

For the PTM family, the determination or selection the number K of the most appropriate latent topics is extremely critical and directly effects the quality of estimated topic set. Up to now, it is still an open-ended problem in topic modeling. While large topic number K means lower descriptor ability of the topic model, as well as intensify the 'forced topic' problem [12]. For being avoid this selection dilemma, topic significance emphasises on evaluating the individual topic quality to serve for the topic groups quality. AlSumait et al. [13] devoted to measuring the distances between three categories 'junk topics' comprised of insignificant word groupings and the legitimate topics. Chang et al. [6] considered the interpretability of a topic as a word intrusion task, and designed topic significance to measure the topic quality in terms of semantic interpretability of the words composed a specific topic. Soon, Lau et al. [14] modified and automated the work of Chang et al. [6] via an improved formulation of Newman [7] based on normalized pointwise mutual information (NPMI). Recently, Chi et al. [15] tried to reranking the top-ranked word set in topic description in order to find the more representative words in topics.

III. incoherence and Bias

In fact, the existing approaches about topic coherence assumed that the topic coherence correlates with the coherence is based on top-N highest ranked word set assigned to the topic, and we let W denote the word set with N words as topic description, W=$(w_1, w_2, \cdots, w_N)$. As an example, Table. I lists the same topic with the represented word set (20 and 200) discovered by LDA on Reuters-10[1] dataset (category: interest) with topic number K is set to 60.

For the first glance, from Table.I, we can intuitively see that the two descriptions of the topic (top-20: W_1, top-200: W_2) are very similar because of some top-ranked words, such as '*credit*, *finance*', and we can easily give 'Finance' category label to the topic. However, we find that this is not the case through further observation. It is obvious that there are lots of non-relevant words appeared in topic description with larger word number being selected, such as in W_2, which makes the topic incoherent. With further investigation, we find that some words are real relevant to the topic, such as '*bonds*, *treasury*' in W_2. Nevertheless, they are generally considered low semantic contribution for the topic representation due to the low-ranking. Intuitively, we can conclude that the representative ability in W_2 works better. In fact, it will choose a small word number to construct W in standard topic modeling, and we find it is inappropriate. Based on these observations, we make an assumption that whether we can promote the rankings of these 'real' words so as to accomplish the representative ability of the topic description, as well as alleviating the dilemma of the word frequency. Furthermore, the subsequent experiments confirm this assumption.

The fact is that not all topics are high coherent, the incoherent topics will intensify the inexpressibility of topic representability. However, in particular, we note that the representative ability of topic becomes will be clarity when being select a large-scale word set. Meanwhile, it will also increase the risk of the redundancy of the word set. That is to say, the word with low probability has representative ability instead [3]. As for the general PTM, the selected word number N is always set to be a constant, and it has been paid less attention in relevant research.

Besides incoherence, we also note that a second problem may play an important role. In particular, the semantic of the topic may be biased towards the meaning of the top ranked word set with high probability, which covering the dominate semantic of the topic. Meanwhile, the low-ranking word set actually acts as a supplement role because of the low probability. In addition, the ambiguity of the word is another influence factor.

From Table. I, we can observe that the topic description in W_2 has two distinct profiles: 'finance' and 'politics', and the description words of them are intertwined in W_2 even though we know that finance and politics are always inseparable distinctly. However, the truth is that it will let the dominate profile to assign the category of the specific topic [16], [17]. Just like

[1]http://kdd.ics.uci.edu/database/reuters21578/reuters21578.html

TABLE I: Topic description examples for the same topic on Reuters-10 (category: interest) (The first line (W_1) is the original top-20 word set, the second line (W_2) is the original top-200 word set, and the third line (W_3) is the new top-200 word set which re-ranking W_2 using our method).

W_1	credit, rates, card, committee, six, finance, Canadian, group, limit, balances, trade, report, Imperial, subcommittee, Citicorp, transaction, legislation, State, previously, American
W_2	credit, finance, move, fee, group, state, It, banking, hopes, financial, committee, Visa, balance, Express, transaction, Citicorp, charges, statement, ct, amount, reduce, responding, Service, subcommittee, OPTIMA, marketing, Chia, Hockin, quarterly, billing, yearly, market-related, allowed, low, example, threatened, represented, stay, workers, returns, news, levels, sees, factor, pressures, Switzerland, Nova, suspended, important, Italys, departments, aid, positions, nation-wide, association, speculated, curve, expenses, AXP, war, features, issuer, two-to-one, link, Braddock, individual, ones, television, overriding, defend, dominant, cardholders, monthly, delighted, Dallas, Quebec, enaction, expired, touch, Taiwan, middle-class, entitlement, recognised, deciding, resumed, talking, Japan, expected, cut,base, lending, rate, institutions, pct, part, recent, pact, major, industrial, nations, Paris, Finance, Ministry, sources, said, based, revision, Trust, Fund, Bureau, Law, approved, parliament, March, abolishing, minimum, interest, deposits, bureau, channels, funds, government, public, works, official, uses, bodies, Development, Bank, Peoples, Corp, corporations, local, enterprises, usually, moves, tandem, long-term, prime, rates, However, impossible, follow, January, legally, set, ministry, abolish, introduce, resolve, problem, stimulate, domestic, economy, Tuesday, bankers, record, effective, February, suggested, reached, agreement, depositors, postal, savings, system, Posts, Telecommunications, welfare, annuity, Health, Welfare, ministries, trying, determine, market, considered, setting, bureaus, deposit, Coupon, new, year, bonds, minus, percentage, points, likeliest, choice, added, Italian, treasury, annual, coupon, payable, two, issues, certificates, CCTs, four, compared
W_3	credit, finance, move, fee, group, state, transaction, banking, hopes, financial, committee, Visa, balances, Hockin, Express, Citicorp, charges, statement, ct, amount, reduce, responding, Service, subcommittee, OPTIMA, marketing, Chia, quarterly, billing, It, yearly, market-related, allowed, low, example, threatened, represented, stay, workers, returns, cardholders, news, levels, sees, factor, pressures, Nova, suspended, important, departments, aid, welfare, annuity, positions, nation-wide, association, speculated, curve, expenses, AXP, war, features, issuer, link, Braddock, individual, ones, television, overriding, defend, dominant, monthly, delighted, Quebec, expired, touch, Switzerland, middle-class, entitlement, recognised, deciding, resumed, talking, expected, cut, base, lending, rate, institutions, pct, part, recent, pact, major, industrial, nations, Paris, Finance, Ministry, sources, said, based, revision, Trust, Fund, Bureau, Law, approved, parliament, March, abolishing, minimum, interest, deposits, bureau, channels, funds, government, public, works, official, uses, bodies, Development, Bank, Peoples, Corp, Taiwan, Italys, two-to-one,corporations, local, enterprises, usually, moves, tandem, long-term, prime, rates, However, impossible, follow, January, legally, set, ministry, abolish, introduce, resolve, problem, stimulate, domestic, economy, Tuesday, bankers, record, effective, February, suggested, reached, agreement, depositors, postal, savings, system, Posts, Telecommunications, Health, Welfare, ministries, trying, determine, market, considered, setting, bureaus, deposit, Coupon, new, year, bonds, minus, percentage, points, likeliest, Japan, choice, added, Italian, treasury, annual, coupon, payable, two, issues, certificates, CCTs, four, compared

the topic in Table. I, the category will be assigned 'Finance'. In this paper, we take an assumption that each specific topic estimated by PTM has a single sematic interpretation but with multi-sematic aspects or profiles, and that is the motivation of our work.

IV. Methodology

In this section, we explain the detail our methodology for increasing representative ability of the topic. Furthermore, we analyse the effects of incoherence or bias that standard PTMs suffered. Once we determine which sematic profile is being covered by each topic, we propose to promote and identify 'good' words within the topic description.

For a given topic description word set, the aim of this paper is to re-rank the word set in order to obtain a better topic representation. Inspired by the diversification scheme in information retrieval research, we focus on selecting the words that are both relevant to the topic and different from the words already selected. The common principle of diversification is to select as diverse results as possible from a given set of retrieved documents [18]. In our case, we aim to select a semantic representation word set to represent the topic with less redundancy among them as much semantic expression aspects as possible, as well as avoiding the ambiguity of the word. We formulate our identification scheme as a re-ranking task, which is similar to the work of [15]. But different from the work of [15], we devote to identifying and refining the quantification the meaningful representative words for topic representation from the diversification point view for avoiding semantic loss of the topic description.

In this section, we elaborate on a general-based WTW (Word Topic Ware) method to identify the meaningful representative word set for increasing representative ability to a specific topic. Thus, we choose the classical implicit diversification approach to realize. Our approach is based on a similar principle to Maximal Marginal Relevance (MMR) [19], which aim is to take both relevance and redundancy into account for the selected documents.

In this paper, each topic is presented to a top-N most probable word set W from the word-topic distribution Φ to represent the topic t. In this paper, we elaborate MMR algorithm to promote the representative ability of the specific word in topic description, to remedy the semantic contribution degree of the specific word for the topic. In the experiments, we iteratively select and order top-ranked N words with new weight scores by using MMR scheme.

The new semantic contribution degree of each word $w_i \in W$ of the specific topic t is calculated by Eq.(1):

$$weight(w_i,t) = \lambda degree(w_i,t) - (1-\lambda) \max_{w_j \in S} sim(w_i,w_j) \quad (1)$$

where S is the selected set of words in W, $degree(w_i,t)$ determines the original representative degree of each word w_i in topic t, and $sim(w_i,w_j)$ determines the similarity between two words pair. λ denotes the interpolation parameter which controls the tradeoff between the relevance and the diversity. In the experiments, we empirically set λ=0.5.

As for a fixed topic number K, we consider that for the specific topic t_m ($m \in [1,K]$), the representative degree value of the word $degree(w_i,t_m)$ should be with high marginal

probability in topic t_m, as well as possessing low marginal probability in other topics, which is calculated by Eq.(2):

$$degree(w_i, t_m) = \phi_{t_m,i} \cdot log \frac{\phi_{t_m,i}}{K\sqrt{\prod_{k=1}^{K} \phi_{k,i}}} \quad (2)$$

where $sim(w_i,w_j)$ denotes the similarity between each node pair, and we use Word2vec [2] to accomplish it.

V. Experiments

In this section, we present the evaluation results that we obtained by applying our proposed framework WTW.

A. Datasets

In this section, we evaluate our topic representation scheme on three large widely datasets: 20NG-bydate [3], Reuter-10 and OHSUMED87-91 [4]. we do use offline topic model so that we can easily extend our work on large collections and avoid the challenge of choosing the topic number.

- **20NG-bydate**: It is a widely used dataset for text classification research. It is highly balanced since each category has about 1000 texts. We use the *bydate* version of this dataset with a total of 18,846 articles that are organized into twenty different categories. This version has been divided into training (60%) and test (40%) set, respectively, and we follow this in the experiment. In addition, we keep the text contained in *Content* field for topic modeling.
- **Reuters-10**: It is another benchmark dataset typically used in the research field of text classification. It contains 21,578 documents in 135 categories. But this dataset is very imbalanced and the variation of category size is quite large. Hence, in the experiment, we left the documents belonging to merely one category and use the 10 largest categories in the dataset (Reuters-10) with a total of 7,285 documents. We use the standard split, 5,228 documents is used as train set and 2,057 documents is used as test set, respectively. In the experiment, we use the *BODY* field for topic modeling.
- **OHSUMED87-91**: It is a widely used dataset for text retrieval and text classification research. It contains five years (1987-1991) relatively short abstracts of references from medical journals in the MEDLINE database with 348,566 documents. In the experiment, we use the *abstract* field for topic modeling and we manually eliminate the invalid documents and left 233,445 documents in fact. We select 119,828 documents used as test and the rest of 113,617 documents used as train set.

For all datasets, we removed HTML tags, stop words, rare words and the word with length less than two or occur in less than five documents. In addition, in 20NG-bydate, the *Content* field of some documents are empty, and we cull these documents manually in the experiment. Table. II gives the detailed statistics information of three datasets.

[2]http://code.google.com/p/word2vec

[3]http://qwone.com/~jason/20Newsgroups/

[4]http://mlr.cs.umass.edu/ml/machine-learning-databases/ohsumed/

TABLE II: Statistics of the datasets.

Dataset	train word number	test word number
20NG-bydate	49,446	34,913
Reuters-10	33,340	22,880
OHSUMED87-91	402,058	441,400

B. Experimental Setting

In order to evaluate our approach, we require a topic model. We apply directly standard LDA to obtain the initial topic description results from each dataset. In the experiments, we use Gibbs sampler to generate the topic-word distributions Φ, and the iteration number of Gibbs sampler adopt a fixed value 1000. During modeling training, the Dirichlet hyperparameters α and β are set to 0.1 and 0.01, respectively. For each topic t be denoted as a list with top-ranked N words, and the value of N ranges from 10 to 400, step is set to 5.

C. Results and Analysis

Topic coherence metric is a popular automatical metric to evaluate the coherence of the topics learnt by topic models. However, it is unfit for our work due to the word-frequency essential peculiarity of the language being used. In order to analysis how effect of the topic representative ability is, with the increasing of the topic representation word number N being selected, we take the *perplexity* to evaluate the performance of topic representation.

As shown in Fig. 1, we exhibit the lowest value of *perplexity* comparisons (top-N 20 and top-N 200) on three datasets.

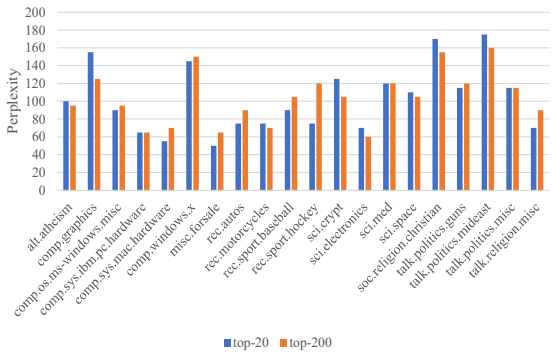
From Fig. 1, we can observe that the lowest value of perplexity has fluctuation to some extent with the selected word number increasing. Experimental results shows that the topic representative ability is instable due to the scale of the top-ranked word set. It is the proof of the influence of word frequency on text modeling. Thus, we can conclude that it is an objective existence phenomenon in general PTMs, and it is also proved that the necessity and significance for increasing the representative ability to topic representation.

In the third line of Table. I shows the new topic description W_3 for the same topic examples. For the first glance, the description of W_3 is similar to W_2, that is to say, they indeed depict the same semantic.

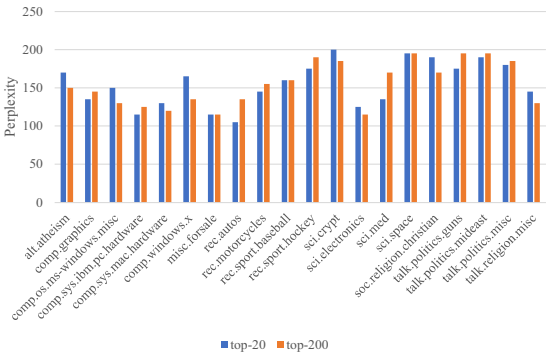
On the one hand, when we make a detailed observation, we find that there are some words, such as '*transaction*, *Hockin*', appeared in W_3 with higher rankings compared to W_2, which makes the semantic representative ability of W_3 better than W_2.

On the other hand, we find that the word with high frequency, such as the word '*It*' in W_2, descend the rankings, i.e, its semantic contribution degree is descend because the little relevance with the selected word set, as well as descending the influence of its word frequency.

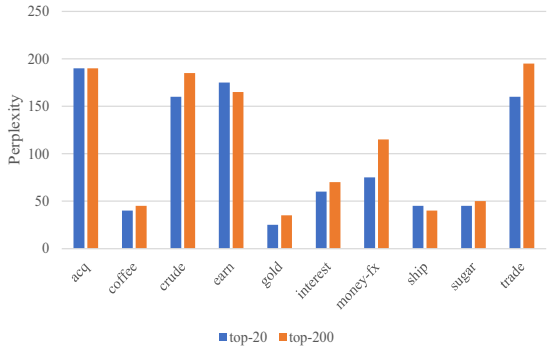
Furthermore, in the experiments, we find that with the value of N ascending, the lower-ranked word set has little semantic contribution for topic representation even though they are ranked using our method, and the representative ability of



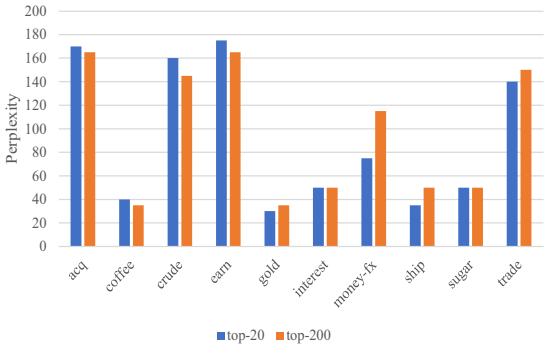
(a) Comparison on 20NG-bydate-Test



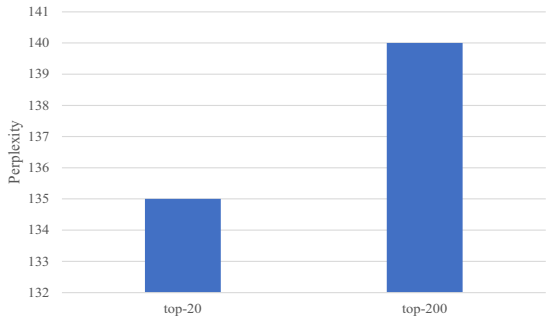
(b) Comparison on 20NG-bydate-Train



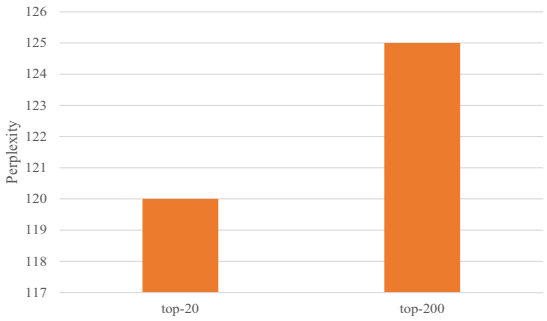
(c) Comparison on Reuters-10-Test



(d) Comparison on Reuters-10-Train



(e) Comparison on OHSUMED87-91-Test



(f) Comparison on OHSUMED87-91-Train

Fig. 1: Examples of perplexity value comparisons on three datasets.

the topic will be confused sometimes. We also find that the value of N is highly depend on the quality of the dataset. If the category of the dataset is relatively distinct, the influence of the N value for the topic representative ability is slight, such as 20NG-bydate and Reuters-10. On the contrary, as for OHSUMED, the influence of the N value for the topic representative ability is volatile.

VI. Conclusion

In this paper, we focus on screening out the more representative words for topic representation. We investigate a re-ranking method WTW to evaluate the representative ability of words over different topics. This work will enhance the ability to support fine grained topics representation for text content mining tasks.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grant No. 61866029).

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, mar 2003.

[2] W. Xuerui and M. Andrew, "Topics over time: A non-markov continuous-time model of topical trends," in *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, ser. KDD'06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 424–433.

[3] R. Yan and G. Gao, "Topic analysis by exploring headline information," in *Proceedings of 21st International Conference on Web Information Systems Engineering*, ser. WISE'20, H. Zhisheng, B. Wouter, W. Hua, Z. Rui, and Z. Yanchun, Eds., vol. 12343. Chem: Springer, oct 2020, pp. 129–142.

[4] Q. Chen, X. Guo, and H. Bai, "Semantic-based topic detection using markov dcision processes," *Neurocomputing*, vol. 242, pp. 40–50, jun 2017.

[5] W. H. M., M. Iain, S. Ruslan, and M. David, "Evaluation methods for topic models," ser. ICML'2009, vol. 382. New York, NY, USA: Association for Computing Machinery, jan 2009, pp. 1105–1112.

[6] C. Jonathan, B.-G. Jordan, G. Sean, W. Chong, and B. D. M., "Reading tea leaves: How humans interpret topic models," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. Red Hook, NY, USA: Curran Associates Inc., 2009, pp. 288–296.

[7] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic evaluation of topic coherence," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. HLT '10. USA: Association for Computational Linguistics, 2010, pp. 100–108.

[8] D. Newman, E. V. Bonilla, and W. Buntine, "Improving topic coherence with regularized topic models," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., 2011, pp. 496–504.

[9] M. David, W. H. M., T. Edmund, L. Miriam, and M. Andrew, "Optimizing semantic coherence in topic models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. USA: Association for Computational Linguistics, 2011, pp. 262–272.

[10] D. O'Callaghan, D. Greene, J. Carthy, and P. Cunningham, "An analysis of the coherence of descriptors in topic modeling," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5645–5657, 2015.

[11] D. Korenčić, S. Ristov, and J. Šnajder, "Document-based topic coherence measures for news media text," *Expert Systems with Applications*, vol. 114, pp. 357–373, 2018.

[12] X. Li, J. Ouyang, Y. Lu, X. Zhou, and T. Tian, "Group topic model: Organizing topics into groups," *Information Retrieval*, vol. 18, no. 1, pp. 1–25, feb 2015.

[13] A. Loulwah, B. Daniel, G. James, and D. Carlotta, "Topic significance ranking of lda generative models," in *Proceedings of Joint European Conference on Machine Learning (ECML) European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 67–82.

[14] J. H. Lau, D. Newman, and T. Baldwin, "Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality," in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, jan 2014, pp. 530–539.

[15] J. Chi, J. Ouyang, C. Li, X. Dong, X. Li, and X. Wang, "Topic representation: Finding more representative words in topic models," *Pattern Recognition Letters*, vol. 123, pp. 53–60, 2019.

[16] R. Yan, Q. Chen, and G. Gao, "Dataless text classification with pseudo topic representation," in *Proceedings of 32nd International Conference on Tools with Artificial Intelligence*, ser. ICTAI'20, nov 2020, pp. 1255–1259.

[17] D. Zha and C. Li, "Multi-label dataless text classification with topic modeling," *Knowledge and Information Systems*, vol. 61, no. 1, pp. 137–160, oct 2019.

[18] R. Santos, C. Macdonald, and I. Ounis, "Search result diversification," *Foundations and Trends in Information Retrieval*, vol. 9, no. 1, pp. 1–90, mar 2015.

[19] J. Carbonell and J. Stewart, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st International ACM Conference on Research and Development in Information Retrieval*, ser. SIGIR'98, Melbourne, Australia, aug 1998, pp. 335–336.

Exploring MMSE Score Prediction Model Based on Spontaneous Speech

Li Sun
School of Computer Science and Technology
Donghua University
Shanghai, China

Jieyuan Zheng
School of Computer Science and Technology
Donghua University
Shanghai, China
zjy123h@163.com

Jiyun Li
School of Computer Science and Technology
Donghua University
Shanghai, China

Chen Qian
School of Computer Science and Technology
Donghua University
Shanghai, China

***Abstract*—The Mini Mental State Examination, referred to as MMSE, is a screening tool for cognitive dysfunction in the elderly, and it is also one of the most influential screening tools for cognitive impairment. It is usually managed by a well-trained doctor, but this is time-consuming and expensive. An effective method is to detect whether cognitive function has declined through the conversation between them. From the perspective of acoustics and linguistics, using 108 subjects provided by the Alzheimer's Dementia Recognition through Spontaneous Speech (ADReSS) 2020 Challenge, using speech to predict the MMSE score, the acoustic Root Mean Squared Error (RMSE) is 5.49. The RMSE in linguistics is 4.51. Integrating the acoustic model and the linguistic model, and assigning different weight ratios to their final predicted scores, the RMSE is 4.18.**

***Index Terms*—Alzheimer's disease, acoustic features, linguistic features, MMSE**

I. Introduction

Alzheimer's Disease (AD), also known as Alzheimer's, is a neurodegenerative disease. According to epidemiological studies, the incidence of AD increases with age, about 5% of people over 65 years old, and up to 20% of people over 85 years old. According to statistics from Western countries [1], it is estimated that between 2000 and 2050, the population over 65 will triple, which will undoubtedly greatly increase the burden on families and the country. Because it is an irreversible disease, drug treatment may temporarily change the symptoms of the disease, but it cannot reverse its progress. For these reasons, there is an increasing need for this additional, non-invasive detection tool to enable preliminary identification of AD at an early stage.

At present, the more popular detection methods are to use Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), but this is undoubtedly more expensive. In the process of cognitive decline, the appearance of language barriers [2] is an important sign, which includes naming [2], difficulty in finding words, repetition, and improper use of pronouns [3]. This makes it possible to use speech to evaluate the AD process.

Cognitive assessments are often used for clinical validation, such as the Mini-Mental Status Examination (MMSE) [4]. Although simple to administer MMSE, it is burdensome for subjects and may also be influenced by various demographic factors [5]. Preliminary evidence [6] shows that automated methods can predict MMSE scores from open communication. Based on this, the main contributions of this paper are as follows. First, uses opensmile-3.0 to extract acoustic features, including ComParE16, emobase, eGeMAPS and Is09-13, and put the extracted features into the acoustic model. Second, there are two types of linguistic features. The first is to use BERT [7] to extract sentence vectors; the second is to use n-grams to vectorize text, combine psycholinguistic features, and put them into machine learning models. Third, comprehensively consider the acoustic model and the linguistic model, and fuse the models at the decision-making level.

II. Related work

In recent years, people have paid more and more attention to speech and language disorders in AD. However, most of the work is focused on dementia classification tasks [8, 9], rather than more detailed prediction of MMSE scores [10]. Aparna Balagopalan [11] demonstrated the use of domain knowledge-based methods to extract linguistic features from text and acoustic features from corresponding audio files, and combine two regression models, namely linear model and ridge regression model. The RMSE obtained is 4.56. Morteza Rohanian [12] and others used the LSTM with gating multi-modal fusion model, combined with multi-modal features, and the RMSE obtained was 4.54. Utkarsh Sarawgi [13] used transfer learning and ensemble models and got an RMSE of 4.60.

Although the RMSE of these papers is lower than the baseline, there is also a problem, that is, the impact of acoustic and linguistic models on the final results is not fully considered. Especially after the prediction results of the two models are obtained, the respective influences on the final results are comprehensively considered, and different weights are given respectively when the decision-making layer is fused.

III. Dataset and Features

A. Overview of the Dataset

The data set of this paper comes from the ADReSS Challenge [6], the subjects provided are theft pictures, which are provided by the Boston Diagnostic Aphasia Exam [14, 15].

DOI reference number: 10.18293/SEKE2022-062

During the recording process, the subject asked to describe the content in the picture, there is no time limit (the interviewer may stimulate the subject to add details). The provided .cha file is a manual transcription of the audio, using the CHAT encoding system [16], which contains non-verbal clues such as adding false starts, pauses, discourse markers for word repetition, and incomplete sentences. For the ADReSS Challenge, the original speech is also divided into standardized segments with a maximum length of ten seconds.

The ADReSS challenge data set is a balanced subset consisting of 156 subjects. Each subject provides a speech. Between AD and non-AD, age and gender are evenly distributed. The following two tables (TABLE I and TABLE II) respectively show the basic situation of the training set and test set.

TABLE I
ADReSS Training Set: Basic characteristics of the patients in each group (M=male and F=female)

	AD			Non-AD		
Age	**M**	**F**	**MMSE**	**M**	**F**	**MMSE**
[50,55)	1	0	30.0	1	0	29.0
[55,60)	5	4	16.3	5	4	29.0
[60,65)	3	6	18.3	3	6	29.3
[65,70)	6	10	16.9	6	10	29.1
[70,75)	6	8	15.8	6	8	29.1
[75,80)	3	2	17.2	3	2	28.8
Total	24	30	17.0	24	30	29.1

TABLE II
Characteristics of the ADReSS test set

	AD			Non-AD		
Age	**M**	**F**	**MMSE**	**M**	**F**	**MMSE**
[50,55)	1	0	23.0	1	0	28.0
[55,60)	2	2	18.7	2	2	28.5
[60,65)	1	3	14.7	1	3	28.7
[65,70)	3	4	23.2	3	4	29.4
[70,75)	3	3	17.3	3	3	28.0
[75,80)	1	1	21.5	1	1	30.0
Total	11	13	19.5	11	13	28.8

B. Acoustic Features

The ComParE16 feature set [17] is extracted using opensmile-3.0. The feature set contains 6373 static features, which are obtained by calculating various functions on LLD (low-level descriptors, LLD). LLD includes logarithmic harmonic noise ratio, voice quality characteristics, F0 Viterbi smoothing, spectral harmonics and psychoacoustic spectral sharpness. This feature set encodes human speech and has been used as an important non-invasive marker for AD detection. We remove the mean and normalize the variance of the obtained feature set. Standard deviation standardization makes the processed data conform to the standard normal distribution, that is, the mean is 0 and the standard deviation is 1. The transformation function is as follows:

$$X^* = \frac{\chi - \mu}{\sigma} \tag{1}$$

Where μ is the mean of all sample data, and σ is the standard deviation of all sample data.

In addition to ComParE16, features of emobase, eGeMAPS, and Is09-13 are also extracted for comparison experiments.

C. Linguistic Features

The ADReSS data set provides a corresponding .cha file for each subject, which contains the conversation between the interviewer and the subject (beginning with *INV and *PAR, respectively). First, extract the subject's speech fragments according to certain rules, and then use TF-IDF (term frequency—inverse document frequency) for the text. The calculation formula is as follows:

$$TFIDF = TF \times \frac{1}{DF} \tag{2}$$

Where TF is the term frequency in the text, and DF is the number of documents containing the current term. In addition to considering the frequency of a vocabulary in the text, it also pays attention to the number of all texts that contain this vocabulary. This can reduce the impact of high-frequency meaningless vocabulary and dig out more meaningful features.

For psycholinguistic features, four classic psycholinguistic attributes (age of acquisition, concreteness, familiarity, and imageability) and emotion scores are considered. They are obtained from the Medical Research Council (MRC) psychology database and Natural Language Toolkit (NLTK) respectively.

Pre-training of Deep Bidirectional Transformers for Language Understanding (BERT), this model mainly uses the Encoder structure of Transformer, but the model structure is deeper than Transformer. The Transformer Encoder contains 6 Encoder blocks, the BERT-base model contains 12 Encoder blocks, and the BERT-large model contains 24 Encoder blocks.

Through the pre-training model BERT, the text of each subject is converted into a 768-dimensional sentence vector, and the obtained sentence vector is normalized.

IV. Experiment

A. Acoustic Model

Multilayer Perceptron (MLP) is a neural network with a forward structure that maps a set of input vectors to a set of output vectors. MLP can be regarded as a directed graph, composed of multiple node layers, and each layer is fully connected to the next layer. Except for the input node, each node is a neuron with a nonlinear activation function. MLP is the promotion of traditional perceptrons, which overcomes the weakness that traditional perceptrons cannot recognize linearly inseparable data.

For the acoustic model, as shown in Fig. 1, this article uses opensmile-3.0 to extract ComParE16, emobase, eGeMAPS and is09-13 features, and then put them into MLP, which contains five fully connected layers, and uses L1 regularization in the layer. Add the Dropout layer to the second layer and the penultimate layer, randomly remove some neurons in the network, thereby reducing the dependence on the weight of w, so as to reduce the effect of fitting. The activation function

sigmoid is added to the fully connected layer, and the ReLu activation function (max_value = 30) is added to the output layer to predict the MMSE score.

During the training process, we set epsilon to 1e-07, learning-rate to 0.01, batch_size to 16, epochs to 2000, and loss and metrics to use Mean Square Error (MSE).

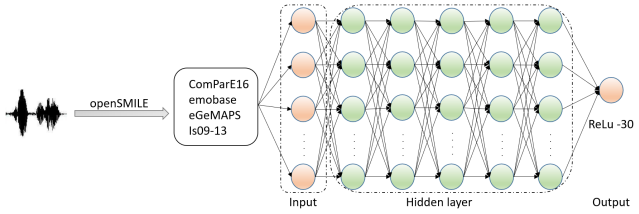


Fig. 1. Our proposed for acoustic model.

B. Linguistic Model

For the linguistic model, as shown in Fig. 2, there are two types. The first is to use the pre-trained BERT model to extract sentence vectors and standardize the obtained features. Taking into account the large difference in MMSE scores, the corresponding score of each subject is divided by 30, and then standardized, combined with the machine learning model (due to the low dimensionality of the feature, machine learning is used), better results can be obtained on ridge regression. During the training process using the ridge regression model, we set alphas=numpy.linspace(1,0.05), store_cv_values=True. The second is to use lexical features to combine emotional factors. First, use TfidVectorizer to extract syntactic features, and then obtain emotional scores from NLTK (Natural Language Toolkit). The obtained features are selected using random forest regression algorithm. The processing of the MMSE score is the same as above, and finally combined with the SVR model to get a better result. In the process of using SVR training, we set the kernel to poly, c to 100, gamma to scale, degree to 3, epsilon to 0.01, and coef0 to 1.

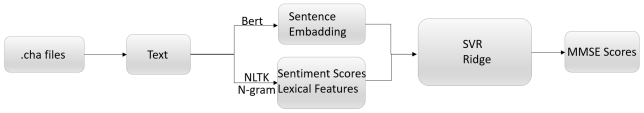


Fig. 2. Our proposed for linguistic model.

C. Fusion of Acoustic and Linguistic Model

Considering the acoustic model and the linguistic model, the results obtained by the acoustic model and the linguistic model are combined. Considering that the effect of the linguistic model is better than that of the acoustics, the weight of the acoustic model is appropriately reduced. In the experiment, the weights were evenly distributed first, and the results obtained by the linguistic model and the acoustic model were multiplied by a weight of 0.5, and then according to the weight of 0.1, the weight of the results obtained by the linguistic model was increased, and the weight of the results obtained by the acoustic model was decreased.After many experiments, it is found that multiplying the result obtained by the acoustic model by a weight of 0.3 and the result obtained by the linguistic model by 0.7, the optimal result of the combination of the two models can be obtained.

D. Experimental Environment

The experiment proceeded from three different perspectives, first using MLP to process the acoustic features, then using machine learning to process the linguistic features, and finally integrating the results obtained from the acoustic and linguistic models, and comparing and analyzing with the baseline.

The environment used in this paper is python3.6, the deep learning framework is tensorflow-based keras framework, the machine learning library is scikit-learn, and the operating system used is windows 10. The experiment set up a five-fold crossover, and used the trained model to predict the test set.

V. Results and Analysis

In order to effectively evaluate the features extracted in this article and the effectiveness of the models adopted, RMSE is proposed as an evaluation index.

Root Mean Squard Error (RMSE) is the square root of the ratio of the square of the deviation between the predicted value and the true value to the number m of the test set. It is used to measure the deviation between the predicted value and the true value. The smaller the value, the better the prediction effect of the model. The specific formula is as follows:

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2} \tag{3}$$

Among them, y_i represents the predicted value, and $\hat{y_i}$ represents the actual value.

In the baseline experiment, it has been proved that the acoustic and linguistic features in spontaneous speech have a certain correlation with the detection of cognitive impairment, and the relevant results have been provided. For the test set, the RMSE of acoustic features is 6.14, and the RMSE of linguistics is 5.21.

This experiment sets up five-fold cross-validation. TABLE III shows the results of acoustic features on the training set and test set. It can be seen that using ComParE16+MLP has the best effect on the test set. The RMSE is 5.49, which is 10% lower than the acoustic baseline. TABLE IV shows the results of linguistic features on the training set and test set. It can be seen that using Lexical+sentiment+SVR performs best on the test set, with an RMSE of 4.51, which is 13% lower than the linguistic baseline. TABLE V shows that the results of acoustic model and linguistic model are assigned weights of 0.3 and 0.7 respectively. It can be seen that the combination of acoustic feature ComParE16 and linguistic feature Lexical+sentiment results in the best result, and the RMSE is 4.18. It is 31.9% lower than the acoustic baseline and 19.8% lower than the linguistic baseline.

TABLE III
THE RESULTS OF ACOUSTIC MODEL ON THE TRAINING SET AND TEST SET

Features	Model	RMSE on train set	RMSE on test set
baseline	-	7.28	6.14
ComParE16	MLP	5.46	**5.49**
emobase	MLP	**4.73**	5.82
eGeMAPS	MLP	5.06	5.96
Is09-13	MLP	5.08	6.28

TABLE IV
THE RESULTS OF LINGUISTIC MODEL ON THE TRAINING SET AND TEST SET

Features	Model	RMSE on train set	RMSE on test set
baseline	-	4.38	5.21
Bert embedding	ridge	4.86	5.37
Lexical+sentiment	SVR	**4.01**	**4.51**

TABLE V
THE RESULTS OF FUSION MODEL ON THE TEST SET

Features	Model	RMSE on test set
ComParE16+ Lexical+sentiment	**MLP+SVR**	**4.18**
ComParE16+Bert embedding	MLP+ridge	4.91
eGeMAPS+ Lexical+sentiment	MLP+SVR	4.20
eGeMAPS +Bert embedding	MLP+ridge	4.93
emobase+ Lexical+sentiment	MLP+SVR	4.40
emobase+Bert embedding	MLP+ridge	4.87
Is09-13+ Lexical+sentiment	MLP+SVR	4.32
Is09-13+Bert embedding	MLP+ridge	4.93

VI. CONCLUSIONS

For the use of speech to predict MMSE scores, there are relatively few research papers in this area. The paper starts from acoustics and linguistics, combined with MLP and machine learning models, and finds that linguistics can provide more information such as pauses, word repetitions, incomplete sentences and emotions, which provide us with strong evidence for predicting MMSE scores.

In the follow-up work, on the one hand, we can also start with linguistics to discover more meaningful features. On the other hand, for acoustic features, we can extract spectrograms such as Spectrongram (Spec), Melspectrongram (Melspec), Mel-Frequency Cepstral Coefficients (MFCC), etc., and combine convolutional neural networks (CNN) to learn two-dimensional features. For speech that cannot be transcribed, the pre-training model wav2vec2.0 can also be used to encode the speech information and modify the downstream output terminal to obtain the expected result.

REFERENCES

[1] OMS, "Es mental health action plan 2013 - 2020," 2013.

[2] J. Reilly, J. Troche, and M. Grossman, *Language Processing in Dementia*. The Handbook of Alzheimer's Disease and Other Dementias, 2011.

[3] D. N. Ripich and B. Y. Terrell, "Patterns of discourse cohesion and coherence in alzheimer's disease." *J Speech Hear Disord*, vol. 53, no. 1, pp. 8–15, 1988.

[4] J. R. Cockrell and M. F. Folstein, "Mini-mental state examination (mmse)." *australian journal of physiotherapy*, vol. 51, no. 3, pp. 689–92, 2005.

[5] R. N. Jones and J. J. Gallo, "Education and sex differences in the mini-mental state examination," *Journals of Gerontology*, no. 6, p. 6.

[6] S. Luz, F. Haider, S. de la Fuente, D. Fromm, and B. MacWhinney, "Alzheimer's dementia recognition through spontaneous speech: The adress challenge," *arXiv preprint arXiv:2004.06833*, 2020.

[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[8] K. C. Fraser, J. A. Meltzer, F. Rudzicz, and P. Garrard, "Linguistic features identify alzheimer's disease in narrative speech," *Journal of Alzheimerš Disease*, vol. 49, no. 2, pp. 407–422, 2015.

[9] A. K?Nig, A. Satt, A. Sorin, R. Hoory, O. Toledo-Ronen, A. Derreumaux, V. Manera, F. Verhey, P. Aalten, and P. H. a. Robert, "Automatic speech analysis for the assessment of patients with predementia and alzheimer's disease," *Alzheimer s Dementia Diagnosis Assessment Disease Monitoring*, vol. 1, no. 1, p. 112–124, 2015.

[10] M. Yancheva, K. Fraser, and F. Rudzicz, "Using linguistic features longitudinally to predict clinical scores for alzheimer's disease and related dementias," in *Slpat: Workshop on Speech Language Processing for Assistive Technologies*, 2015.

[11] A. Balagopalan, B. Eyre, F. Rudzicz, and J. Novikova, "To bert or not to bert: Comparing speech and language-based approaches for alzheimer's disease detection," 2020.

[12] M. Rohanian, J. Hough, and M. Purver, "Multi-modal fusion with gating using audio, lexical and disfluency features for alzheimer's dementia recognition from spontaneous speech," 2021.

[13] U. Sarawgi, W. Zulfikar, N. Soliman, and P. Maes, "Multimodal inductive transfer learning for detection of alzheimer's dementia and its severity," 2020.

[14] ASHAWeb, "Boston diagnostic aphasia examination-third edition (bdae-3)," *Asha*, 2000.

[15] F. Boersma and J. A. Eefsting, "The natural history of alzheimer's disease," *Journal of the American Geriatrics Society*, vol. 44, no. 6, pp. 734–734, 1996.

[16] J. W. Oller and B. Macwhinney, "The childes project: Tools for analyzing talk, 3rd edition, vol 1, transcription format and programs," *Modern Language Journal*, vol. 86, no. 2, pp. 289–290, 2002.

[17] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in opensmile, the munich open-source multimedia feature extractor," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013.

NKind: a model checker for liveness property verification on Lustre programs

Junjie Wei, Qin Li*
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University, Shanghai, China

***Abstract*—Modeling and verification of real-time reactive systems is getting greater concern in industrial field, especially in safety-critical applications. As a representative language for modeling real-time reactive systems, Lustre has been extensively used in the development of control systems in vehicles and aircraft. Existing model checking tools for Lustre like Kind2 and JKind have good support for verifying safety properties, but they lack explicit support for liveness properties. Thus we present NKind, an SMT-based infinite-state model checker, which accepts models and properties written in Lustre and is capable of verifying both safety and liveness properties. NKind is inspired by many existing model checker and adds liveness support based on their common techniques, which provides more flexibility. It is written in Java, providing good compatibility, and lays emphasis on modularity and extensibility. The results and performance of NKind on benchmark examples demonstrate that it is competitive comparing to other existing tools.**

***Index Terms*—Lustre, Model Checking, Liveness Property**

I. Introduction

As one of the most important measures of ensuring that software meets the expected requirements, model checking is becoming increasingly important in the development of modern systems, especially real-time reactive systems. Many industrial standards like DO-178C, EN50128, ISO26262 etc. require to use formal verification in software design and development for high safety assurance, which demonstrate the bright prospect of wide application of model checking tools in industrial field. As a representative language for modelling real-time reactive systems, Lustre [1] has been extensively used in the development of safety-critical systems like avionic systems and power plant monitoring systems.

Kind2 [2] and JKind [3] are two most popular model checking tools for verifying Lustre programs. Kind2 is a multi-engine model checker and lays emphasis on invariant checking. It uses an extension to Lustre as modelling language, and converts the given model into a state transition system. The property is proved by checking that it holds in all reachable states of the system. JKind provides similar functionalities. It mainly focuses on post-processing and proposes features like inductive validity cores (IVC) and smoothing. JKind and Kind2 support different Lustre language features. For example, JKind lacks the support for automaton structure.

It is worth mentioning that these tools are mostly concentrated on proving safety properties and ignore the liveness properties. This leads to a gap in verifying liveness properties for Lustre. nuXmv [4] is a model checker capable of both finite-state and infinite-state systems. As the successor of NuSmv, it reads models in SMV format extended with infinite-state support. Although nuXmv support liveness property checking, currently there is no available way to make it support Lustre models directly.

The main contributions of the paper can be summarized as follows:

- We present NKind, a model checker for Lustre supporting the verification of liveness properties which existing tools for Lustre do not support.
- When verifying liveness properties, the performance gap between NKind and mainstream tools is not significant.

In this paper, we present NKind, an SMT-based infinite-state model checking tool, which is mainly used for proving or disproving properties of synchronous reactive system models written in Lustre. NKind mainly relies on the powerful SMT-solver Z3 [5] to validate or invalidate the properties. For properties that are proved to be invalid, a counterexample will be returned. NKind is inspired by several existing model checkers for Lustre like Kind2 and JKind, and uses similar architecture and techniques. In the mean time, it provides enough extensibility and makes it rather easy to support new features. NKind is written in Java, which offers better multi-platform compatibility and is easy to be integrated to a larger framework as a model checking service provider. To fill the gap in liveness property checking in Lustre, in addition to safety properties, NKind has the capability to verify liveness properties which can have finite-time violations but will finally hold forever. NKind is free to be used for research and evaluation purposes and can be downloaded from https://nkindmodelchecker.github.io.

The rest of the paper is organized as follows. Section II briefly introduces some preliminary concepts involved in NKind. Section III describes the design architecture of NKind and introduces the main algorithms that are used for verification. Section IV provides the capabilities of NKind and is emphasized on the liveness extension. Section V conducts performance benchmarks for NKind and makes comparison with other existing model checking tools Finally, conclusion and future work is given in Section VI .

*Corresponding author: Qin Li (qli@sei.ecnu.edu.cn)

DOI reference number: 10.18293/SEKE2022-089

II. Preliminaries

A. Lustre language

Real-time reactive systems refer to the systems that continuously accept input and react to the environment in a timely manner. Since the inputs are constantly changing and the systems are expected to respond quickly to the inputs [6], synchronous languages were designed to effectively describe reactive systems. Lustre is a synchronous dataflow language which is widely used in safety-critical control systems like vehicles and aircraft.

Different from imperative languages, dataflow languages focus on data and represent them as infinite sequences of values, i.e. dataflows. This fits well with the usage scenario of real-time reactive systems, which in many cases need to read data from multiple sensors at a fixed frequency as inputs and then calculate the outputs, and the dataflow model can represent this behaviour well. Each dataflow is associated with a clock, and the specific value of a dataflow at a clock time can be uniquely determined by using the clock value as index.

Lustre use a node as a minimal functional module, which has a finite group of inputs and outputs as interface and allows local flow to save its internal state. Functionally speaking, a Lustre node can be regarded as a mapping from an input set to an output set [6], and the outputs are calculated from the current and previous input/output or local flows according to the flow definition. Readers can refer to [1] for detailed grammar of Lustre language.

B. L2SIA-WFR

In comparison with safety properties which have counterexamples of finite length, liveness properties often have counterexamples of infinite length, making it more difficult to verify liveness properties to a large extent. Algorithms like liveness-to-safety [7] and k-Liveness [8] were proposed to solve this problem, but these solutions were mainly restricted in finite-state systems, which were not sufficient to work in infinite-state systems like Lustre. Then an extended version of liveness-to-safety called L2SIA-WFR [9] was presented to handle the infinite state space.

A counterexample of liveness property in the form of $FG\ \neg p$ is often lasso-shaped, which consists of a path starting from the initial state (i.e. stem) and a loop containing at least one state satisfying p. So liveness-to-safety tries to prove the absence of a lasso-shape counterexample as an invariant by ensuring there is no loop paths violating the property. L2SIA-WFR first extend the algorithm by using implicit abstraction. It uses a set of assignments to the predicates to identify multiple concrete states, therefore abstracting the infinite state space to finite state space. Like the idea of CEGAR [10], if spurious counterexamples are found, they will be used to refine the abstraction for next iteration by adding extracted predicates from counterexamples to the predicate set. In order to handle the situations that an abstract loop can be executed finite times, which will not violate the property but prevent the algorithm from terminating, well-founded relations are calculated as a termination proof. In the cases where well-founded relations are available, these relations provides more information of the model for better refining the abstraction and lead to a better performance.

III. NKind Architecture

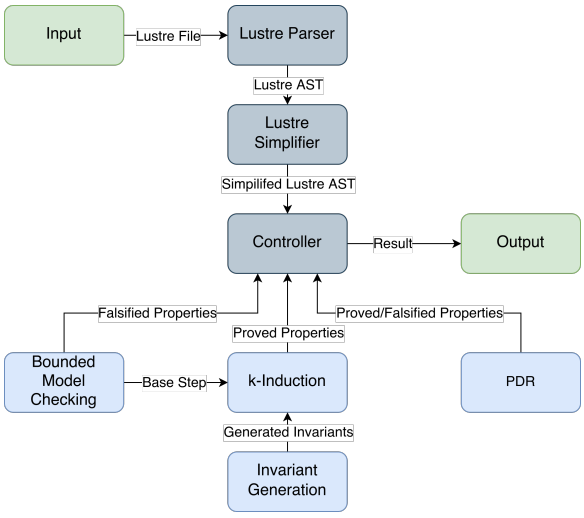


Fig. 1: NKind architecture

The overall architecture of NKind is shown in Fig. 1. It consists of Lustre parser, simplifier, and a controller equipped with several verification engines.

Since there are different dialects of Lustre with different syntax, for scalability reasons, an improved version of visitor pattern is used in the design which provides more flexibility to handle syntactic structures of Lustre and therefore makes it easy to extend functionality.

The traditional version of Lustre, namely Lustre v4, consist of a set of elements which is basic enough and could be considered as Core Lustre. Many features added in more recent versions or dialects can be translated to Core Lustre. For instance, automaton structure, which is available in Kind2 and SCADE [11], introduces a concept similar to state machine and provides the ability to change the behaviour pattern based on external inputs or internal events, allowing a dataflow to have multiple definition in different states. However, such structure can be simulated by converting each state into corresponding nodes without changing the semantics. With this in mind, the Lustre simplifier is used to translate all the complex structure in the given Lustre program before it is converted to transition system and make it possible to leave the process of making transition system unchanged when adding support for new features. It is suitable for implementing syntactic sugar like array iterator introduced in SCADE.

Like several existing model checking tools for Lustre which rely on the expressivity and reasoning capabilities of modern SMT solvers, NKind converts input Lustre program into a transition system with the same semantics in the form of SMT formula. As the core representation of the given model, the

transition system is then handed over to solving engines to verify the properties.

NKind follows the practice of many model checkers and uses a set of solving engines which run in parallel for verification. The solving engines of NKind are mainly composed of Bounded Model Checking (BMC), k-Induction, Invariant Generation and Property Directed Reachability (PDR, or IC3).

1) BMC [12] engine checks for counterexamples by unrolling transition relation T step by step. It also provides the proof of base step for k-Induction engine in the mean time.
2) k-Induction [13] is the enhanced form of normal induction. It tries to find a value of k such that the property p holds for all states reachable from initial state I in first k steps (base step) and is preserved by continuous transitions of length k (induction step), i.e. $\forall i \leq k \cdot I \wedge T_0 \wedge T_1 \wedge \cdots \wedge T_k \Rightarrow p_k$ and $\forall n \geq 0 \cdot T_n \wedge p_n \wedge T_{n+1} \wedge p_{n+1} \wedge \cdots \wedge T_{n+k} \Rightarrow p_{n+k}$. If such k exists, it follows inductively that the property holds in all reachable states.
3) Invariant Generation automatically generates some validated auxiliary invariants based on predefined invariant templates [14] according to the given transition relations and are proved by k-Induction. The generated invariants are mainly used for helping the verification process of k-Induction engine in case the given property is not k-inductive.
4) PDR [15] is based on an idea similar to CEGAR to make an over-approximation of the property and iteratively strengthen the approximation until it becomes inductive or meets a counterexample. The original PDR algorithm is only capable of handling finite-state problem, and in order to make it work in infinite-state system, implicit abstraction proposed in [16] to abstract the states into finite ones. The abstraction itself is refined by extracting new predicates from the Craig interpolants of spurious counterexamples.

Once a property is proved or disproved, other engines will be informed to make use of the result. If the verification process is interrupted or the backend SMT solver encounters an error, the property will be marked unknown and returned to user.

IV. Main Features

A. Safety Property

One of the main functionalities of NKind is to verify safety properties of reactive system modelled in Lustre language. Like Kind2, JKind or other Lustre model checkers, NKind attempts to prove that the given properties are invariants in the given system with a set of model checking engines which are described in the previous section, and tries to give a counterexample in case of failure.

B. Liveness Property

Apart from the traditional safety property checking, NKind also introduces some new techniques for liveness property checking, which, to the best of our knowledge, makes NKind the first model checker for Lustre that support liveness properties. In general, if we use Linear Temporal Logic (LTL) to summarize the property that NKind is able to handle, properties in form of $G\ p$ are supported to enable traditional safety property checking. In addition, properties in form of $FG\ p$ are also supported due to the liveness extension. In other words, apart from checking properties that hold forever, properties which have finite-time violations can be allowed as long as the properties will finally holds forever.

1) Liveness Usage: With the liveness support, Lustre becomes more expressive when specifying property. For instance,

```
node main () returns (x: int; f: bool);
var
  N : int;
  e1 : bool;
let
  N = (20 -> pre (N));
  x = (0 -> (pre (x) + 1));
  f = true -> ((pre (x) < 1) or (pre (x) > pre (N)));
  e1 = f;
  --%MAIN;
  --%PROPERTY Live e1;
tel;
```

Fig. 2: Example Lustre program with liveness property

the lustre program shown in Fig. 2 mainly contains three dataflow.

1) N is a constant flow with value 20.
2) x self-increases by 1 per cycle.
3) f indicates the property that either the value of x in the previous cycle is less than 1 or greater than that of N.

It is clear that the property is violated when $1 \leq i \leq 20$. By specifying the property type using keyword "Live", NKind can be informed to not simply use invariant proving techniques but to encode the property first. As is mentioned in the previous section, the encoded property is to prove the absence of a loop continuously violating the original property. In this case, after x is increased to 21, x will never be less than N and thus proving the original property.

As a result, the constraints of traditional safety properties can be loosen, and it will be easier to specify qualitative property without giving an explicit value. For example, the program listed in Fig. 3 describes a system where y will finally catch up with x. If we use safety property to specify this property, an explicit gap is needed and thus a counter is introduced. However, we can write a more general property if we use the liveness extension which is displayed in Fig. 4.

It will also be suitable for specifying the system that will enter a stable state after several temporary transition caused by input events from sensors, which is common in various industrial control systems.

2) Liveness implementation: As is mentioned before, the verification process of NKind revolves around the state transition system converted from the given Lustre program. L2SIA-WFR algorithm [9] provides a method to encode liveness properties as safety properties at the level of state transition system.

```
node main() returns(x: int; y: int; f: bool);
var
    e1: bool;
    counter: int;
let
    x = 20 -> (pre x) + 1;
    y = 0 -> (pre y) + 2;
    f = false -> pre x < pre y;
    counter = 0 -> pre counter + 1;
    e1 = (counter > 21) => f;
    --%MAIN;
    --%PROPERTY Safe e1;
tel;
```

Fig. 3: Specifying property with safety property

```
node main() returns(x: int; y: int; f: bool);
var
    e1: bool;
let
    x = 20 -> (pre x) + 1;
    y = 0 -> (pre y) + 2;
    f = false -> pre x < pre y;
    e1 = f;
    --%MAIN;
    --%PROPERTY Live e1;
tel;
```

Fig. 4: Specifying property with liveness property

This commonality in structure makes it possible to introduce support for liveness property in the original framework.

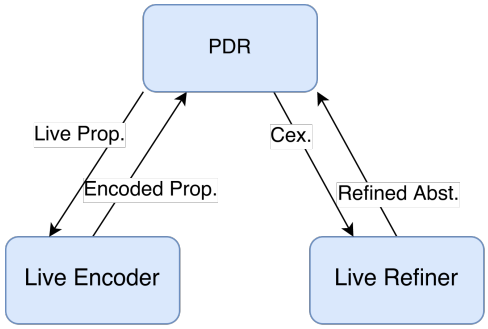


Fig. 5: Embedding liveness extension to PDR process

Since the method monotonically strengthens the original transition system and is a good complement to PDR process [9], the liveness extension in NKind is embedded in PDR engine like Fig. 5 and mainly refers to ic3ia [17], an open-source implementation of the L2SIA-WFR algorithm.

More specifically, a liveness property is first encoded as a safety property using L2SIA algorithm and then put into PDR process. If a counterexample is found and normal refinement of the PDR process failed to make more precise abstraction, Live Refiner will try more liveness-specific methods to make refinement like proving the spuriousness of the counterexample by unrolling the transition relation or attempting to get new ranking relations. Readers can refer to [9] for algorithm details.

Since abstraction refinement in PDR process depends on the calculation of Craig interpolation, an SMT solver which supports such calculation is needed as the backend of the algorithm. For example, JKind chooses to embed SMTInterpol [18], a solver which is devoted to produce interpolants. However, as the input constraints becoming bigger and more complex, this solver may encounter performance degradation. This shortcoming becomes more significant in the liveness extension because the transition system is strengthened monotonically by adding more constraints due to the design of the algorithm, which may lead to variable explosion. Taking the simple Lustre program shown in Fig. 2 as an example, after being translated into transition system, it contains 35 symbols. But after the first liveness encoding, it grows to 284 symbols. Such increment will obviously impose a greater burden on the solver.

Due to the lack of SMT solvers supporting interpolation, it is not easy to simply switch to another solver with higher performance. However, we mentioned the fact that the dependence on interpolation is limited to the part of abstract refinement, which inspired us to use another efficient solver in the main framework of PDR process. Referring to PDR implementation proposed in [19], we use z3 [5] as the main backend SMT solver, and leave the calculation to SMTInterpol only when interpolation is needed by doing a bi-directional conversion between the two solver.

V. Experimental Evaluation

In this section, we evaluate NKind with both safety and liveness benchmarks. The experiments were all conducted on a 64-bit Linux machine running Ubuntu 20.04 with a 20-core Intel Core i9-10900K processor and 32GB of memory.

A. Safety Evaluation

We use a test suite containing 864 Lustre programs from Kind [20], which was also used in the benchmark of Kind2. Two existing and mature tools, Kind2 and JKind, were tested together as a comparison group. The default options for each tool were used and the timeout threshold was set to 300 seconds for each problem.

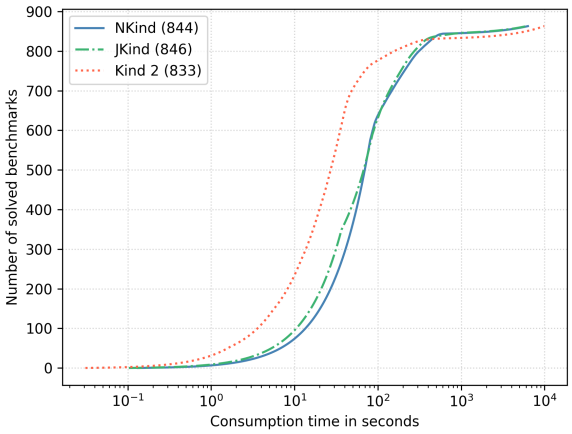


Fig. 6: Verification results on safety property benchmark

The benchmark result is shown in Fig. 6 and the numbers in parentheses represent how many problems were solved in the benchmark. Although there are about 20 programs that cannot be proved or disproved within 5 minutes, the result shows that NKind was capable of proving or invalidating most of the problems in the test suite and had a similar performance to JKind. Relatively speaking, although NKind takes more time when verifying small problem due to the JVM start time, it still has a competitive performance.

B. Liveness Evaluation

Since infinite-state liveness property checking is not as widely supported as safety property is, we first turned to the benchmarks provided together with ic3ia [9]. However, as we have mentioned before, ic3ia accepts a more general form of transition system written in its own vmt format, an extension of the SMT-LIB language. That means not all problems in that benchmark can be translated to a Lustre program with the same semantic, thus making it difficult to accept Lustre program. Due to the lack of effective tools to translate vmt file into Lustre program, we therefore chose 20 tests from the benchmark and converted them in to Lustre programs by hand. Some parameters in the test file were modified to a bigger value to demonstrate the performance of problems in a larger state space.

In order to reflect the effectiveness of the optimizations mentioned in the previous section, a version of NKind using SmtInterpol as backend solver was tested together with the z3 version. Since Kind2 and JKind do not support liveness property, we used ic3ia as a comparison group. The default option and a timeout limit of 300 seconds was used to run the benchmark.

We present some representative test results in detail in Table. I. From the test result, it is clear that the optimization is effective and has a significant performance improvement on most of the test case comparing to the SMTInterpol version. We can see that ic3ia is still the fastest tool in this test, but the gap between ic3ia and NKind is not very significant.

Test name	ic3ia	NKind-z3	NKind-SMTItp
any-down-live	35.77	1.59	25.51
parallel-live	51.68	11.24	32.09
binary-live	0.09	0.8	1.68
piecewise-live	0.11	1.33	2.32
count-nested-live	0.37	1.86	4.42
stabilize-live	5.16	6.57	13.21
count-down-live	1.05	3.81	7.61
swap-dec-live	1.53	3.89	39.37
count-up-to-sym-live	5.84	13.75	3.95
refine_disj_problem	Timeout	Timeout	Timeout

TABLE I: Representative verification results on liveness property benchmark

In order to test with more complex test cases that are closer to the real industrial applications, we leveraged the cases of Kind used as the safety benchmark by converting all the safety properties into liveness properties. Due to the difficulties in conversion from Lustre program to vmt format, we finally use a set of 253 test cases as the benchmark. Fig. 7 shows the benchmark result.

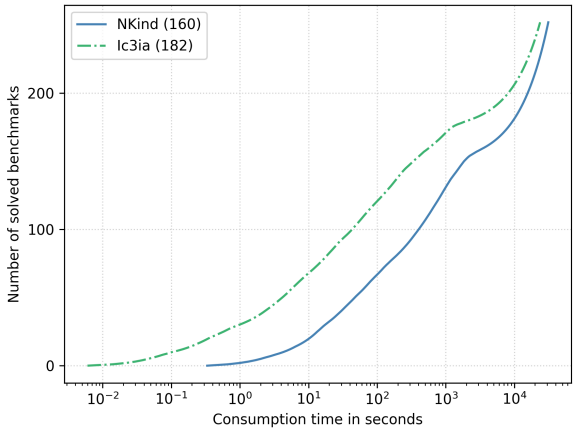


Fig. 7: Verification results on more complex liveness property benchmark

From the benchmark result, NKind is effective in solving liveness properties for Lustre and is able to determine majority of the test cases whether the their properties are valid or not.

However, there is no denying that NKind still has a certain gap compared with the performance of ic3ia. This problem may be caused by the following reason:

1) Due to the use of Java, the JVM start up may consume some time. Since static checking and the translation from Lustre to transition system will be performed before the verification, even small models has a rather long start-up time.

```
node main()
returns
(x : int;f : bool);
var
  N : int;
  e1 : bool;
let
  N = 40 -> pre (N);
  x = 0 -> 1 + pre (x);
  f = false ->  (pre (x) < pre (N));
  e1 = not f;
  --%MAIN;
  --%PROPERTY Live e1;
tel;
```

Fig. 8: Lustre program ”count-up-to-sym-live”

2) Since the liveness extension is mainly based on implicit abstraction version of PDR, we notice that the performance is highly influenced by abstract model. Taking ”count-up-to-sym-live” in the previous benchmark as an example, which is shown in Fig. 8. The model will be verified quickly if the abstraction divide the state space with predicate $x \geq 40$, but will be rather slow if the predicates are $x \leq 1, x \leq 2, \cdots, x \leq 40$. However, we notice that currently the abstraction and the refinement mainly depend on the counterexample returned by the

SMT solver, which is non-deterministic. This may also be the reason why NKind solves less problems. A more guided refinement may be helpful for the performance which need further research.

In general, in spite of such overhead, NKind is capable of handling regular safety properties as well as the liveness ones, which is still believed to be competitive.

VI. Conclusion

In this paper, we presented NKind, an SMT-based model checker for Lustre. Although there are a number of other tools that solve infinite-state model checking problems, NKind integrated their advantages and made the difference. We described its design architecture and functionalities, and emphasized on its extensibility and the support for liveness properties. As far as we know, our liveness extension made NKind the first model checker for Lustre that support both safety and liveness properties, bridging the gap in liveness property checking in Lustre. In the preliminary benchmarks, NKind is proved to be suitable for property verification of industrial control systems, and is rather competitive comparing to the existing model checking tools. Future work includes performing more in-depth optimizations for NKind and improve its performance of model checking by using more guided techniques.

References

[1] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language LUSTRE," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, Sep. 1991, conference Name: Proceedings of the IEEE.

[2] A. Champion, A. Mebsout, C. Sticksel, and C. Tinelli, "The Kind 2 Model Checker," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, S. Chaudhuri and A. Farzan, Eds. Cham: Springer International Publishing, 2016, pp. 510–517.

[3] A. Gacek, J. Backes, M. Whalen, L. Wagner, and E. Ghassabani, "The JKind Model Checker," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, H. Chockler and G. Weissenbacher, Eds. Cham: Springer International Publishing, 2018, pp. 20–27.

[4] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuXmv Symbolic Model Checker," in *Proceedings of the 16th International Conference on Computer Aided Verification - Volume 8559*. Berlin, Heidelberg: Springer-Verlag, Jul. 2014, pp. 334–342. [Online]. Available: https://doi.org/10.1007/978-3-319-08867-9_22

[5] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer, 2008, pp. 337–340.

[6] G. E. Hagen, "Verifying safety properties of lustre programs: an smt-based approach," phd, University of Iowa, USA, 2008, aAI3347220 ISBN-13: 9781109024180.

[7] A. Biere, C. Artho, and V. Schuppan, "Liveness Checking as Safety Checking," *Electronic Notes in Theoretical Computer Science*, vol. 66, pp. 160–177, Dec. 2002.

[8] K. Claessen and N. Sörensson, "A liveness checking algorithm that counts," in *2012 Formal Methods in Computer-Aided Design (FMCAD)*, 2012, pp. 52–59.

[9] J. Daniel, A. Cimatti, A. Griggio, S. Tonetta, and S. Mover, "Infinite-State Liveness-to-Safety via Implicit Abstraction and Well-Founded Relations," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, S. Chaudhuri and A. Farzan, Eds. Cham: Springer International Publishing, 2016, pp. 271–291.

[10] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *J. ACM*, vol. 50, pp. 752–794, Sep. 2003.

[11] G. Berry, "SCADE: Synchronous Design and Validation of Embedded Control Software," in *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, S. Ramesh and P. Sampath, Eds. Dordrecht: Springer Netherlands, 2007, pp. 19–33.

[12] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic Model Checking without BDDs," vol. 4144, Oct. 1999.

[13] M. Sheeran, S. Singh, and G. Stålmarck, "Checking Safety Properties Using Induction and a SAT-Solver," vol. 1954, Nov. 2000, pp. 108–125.

[14] T. Kahsai, P.-L. Garoche, C. Tinelli, and M. Whalen, "Incremental Verification with Mode Variable Invariants in State Machines," vol. 7226, Apr. 2012, pp. 388–402.

[15] A. R. Bradley, "SAT-Based Model Checking without Unrolling," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, R. Jhala and D. Schmidt, Eds. Berlin, Heidelberg: Springer, 2011, pp. 70–87.

[16] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, "Infinite-state invariant checking with IC3 and predicate abstraction," *Formal Methods in System Design*, vol. 49, Dec. 2016.

[17] "IC3ia: IC3 Modulo Theories with Implicit Abstraction." [Online]. Available: https://es-static.fbk.eu/people/griggio/ic3ia/index.html

[18] J. Christ, J. Hoenicke, and A. Nutz, "SMTInterpol: an Interpolating SMT Solver," Jul. 2012.

[19] N. Eén, A. Mishchenko, and R. Brayton, "Efficient implementation of property directed reachability," *2011 Formal Methods in Computer-Aided Design (FMCAD)*, 2011.

[20] G. Hagen and C. Tinelli, "Scaling Up the Formal Verification of Lustre Programs with SMT-Based Techniques," in *2008 Formal Methods in Computer-Aided Design*, 2008, pp. 1–9.

Efficient LTL Model Checking of Deep Reinforcement Learning Systems using Policy Extraction

Peng Jin, Yang Wang, Min Zhang

Shanghai Key Laboratory for Trustworthy Computing, East China Normal University
Shanghai Trusted Industry Internet Software Collaborative Innovation Center
E-mail: 51194501007@stu.ecnu.edu.cn, {ywang,zhangmin}@sei.ecnu.edu.cn

Abstract

Deep Reinforcement Learning (DRL) is a promising technology for solving intractable control tasks. Its applications in safety-critical fields require high-reliability guarantees. However, formal verification of DRL systems is challenging because deep neural networks (DNNs) embedded in the applications are uninterpretable. In this paper, we propose a novel approach to linear temporal logic (LTL) model checking of DRL systems by extracting interpretable policies from DNNs. The extracted policy can retain comparable performance to the original DNN. More importantly, its decision domain is finite and thus directly verifiable against LTL properties using existing model checking techniques. Experimental results on four classic control systems demonstrate the effectiveness of our approach.

1. Introduction

Deep Reinforcement Learning is being utilized heavily to solve diverse problems due to its strength in developing complex control systems [13, 19]. However, new risks emerge with this trend. The reliability of such systems is hard to guarantee because they cannot be formally verified like conventional systems. It becomes one of the significant obstacles to applying DRL in the real world [16, 17].

Three features make it a challenging problem to verifying DRL systems. First, the state space of such control systems is usually infinite and continuous, but most of the model checking-based approaches can only handle finite-state models [20]. Second, the system dynamics are generally nonlinear, which increases the complexity of formal verification [4]. Lastly, DNNs embedded in the systems are inexplicable, restricting the scalability of verification methods [9, 11]. Among them, the black-box nature of DNNs is the crux in defining faithful formal models for DRL systems, which are prerequisites for subsequent verification.

In this paper, we propose a simple but effective method to model checking LTL properties of DRL systems, which bypasses the crux by extracting interpretable policies from DNNs trained with popular DRL algorithms. Then systems driven by the extracted policies can be formally verified using existing model checking techniques. The policy can fulfill two essential requirements. One is that it can offer competitive performance compared to the original DNN. Besides, its decision domain is finite, where the decision-making unit (DMU) is a set of adjacent concrete states. For simplicity, we use DMU to represent the concrete states that it contains. We first discretize the state space of systems into finite DMUs and then determine the action adopted by each DMU to generate the final policy.

Based on extracted policies, we devise an algorithm to transform the DRL system into a finite-state transition system that can be model-checked via off-the-shelf tools such as *Spot* [7] against complex temporal properties. We apply our approach to verify four canonical control systems formally. Experimental results indicate that control systems driven by the extracted policies can be formally modeled and efficiently verified against complex temporal properties.

In summary, this paper makes three major contributions:

1. A novel method for extracting interpretable policies from trained DNNs.
2. An efficient model checking approach for verifying control systems driven by the extracted policies.
3. Four case studies for model checking the temporal properties of four control systems.

Paper organization. Section 2 introduces our policy extraction method. We present a model checking approach for control systems based on the extracted policies in Section 3. Section 4 shows the experimental results. Section 5 discusses related works, and Section 6 concludes the paper.

2. Interpretable Policy Extraction

In this section, we explain the notion of decision-making units (DMUs) used for policy extraction and present a state-space discretization method for generating DMUs and a process of determining corresponding actions for DMUs.

DOI reference number: 10.18293/SEKE2022-029

2.1. Decision-Making Unit

In DRL, policies encoded by DNNs map each concrete state to an optimal action. Therefore, the decision-making unit of such policies is infinite, making it difficult to build formal models for verification. Motivated by the works [2], we use finite axial bounding boxes to define the DMUs of our policy. All DMUs have the same area, and their union forms the state space. More importantly, they do not intersect, which indicates an assumption that concrete states within the DMU adopt the same action.

The assumption above to DMU is reasonable. Firstly, trained DNNs should take the same action for two inputs whose norm distance is close, which reflects the decision robustness [10, 22]. Besides, DRL handles the control systems that require continuous decision-making. Even if the chosen action is not optimal for all concrete states in the DMU, subsequent actions can compensate for the overall policy performance.

2.2. Generation of DMUs

We consider an n-dimensional state space S of the control system. Let L_i and U_i be the lower and upper bounds of the i-th dimension range of S, respectively. We introduce the discretization vector $D \in \mathbb{R}^n$ to divide S into DMUs, where $D = [d_1, \ldots, d_n]$. For the i-th dimension range, it will be discretized into m_i ($= \lfloor \frac{U_i - L_i + d_i}{d_i} \rfloor$) intervals, i.e. $[L_i, L_i + d_i), \ldots, [L_i + m_i d_i - d_i, L_i + m_i d_i)$. Let $R_i = \{L_i, \ldots, L_i + m_i d_i - d_i\}$ be the set of lower bounds of divided intervals in the i-th dimension. Then any DMU can be defined by a pair of two n-dimensional vectors $< [l_1, \ldots, l_n], [d_1, \ldots, d_n] >$, and its corresponding range is $[l_1, l_1 + d_1) \times \ldots \times [l_n, l_n + d_n)$, where $l_i \in R_i$.

2.3. Action Determination

The action determination process is based on the DNNs trained by traditional DRL algorithms. The concrete state originally adopts the action output by the trained DNN. We aim to select the most frequently adopted action by concrete states in the DMU as its optimal action. However, there are infinite concrete states in the DMU, so it is infeasible to calculate precisely the action taken by most concrete states. Motivated by *randomized smoothing* [6] that uses Monte Carlo algorithms to evaluate the class that the trained DNN is most likely to classify for the images close to the input, we sample t concrete states in the DMU and determine its action based on the actions adopted by these concrete states. The sampled concrete state s is generated from n independent uniform distributions, where s_i is sampled from $U[l_i, l_i + d_i)$.

Let Act_a be the most frequently adopted action among t actions. If Act_a appears much more often than other actions, it will be the optimal action for the corresponding DMU. Otherwise, we repeat the sampling process until an explicit optimal action is obtained to avoid achieving a suboptimal action. When the action space of systems is continuous, we directly calculate the average of t actions as the final action of the DMU. In such a case, the action is represented by a vector of real numbers. The action vectors output by the trained DNN are hardly equal, so the actions adopted by sampled concrete states are always different.

Algorithm 1 Sampling-Based Action Determination

Input parameters: DMU **s**, action space AS.
Constant parameters: discretization vector D, sampling count t and statistical significance α.

```
function DETERMINEACTION(s, AS)
    States ← sample t concrete states from s
    Actions ← input States to the trained DNN
    if TYPE(AS) is continuous then
        return AVERAGE(Actions)
    Act_a, Act_b ← top two indices in Actions
    Cnt_a, Cnt_b ← Actions[Act_a], Actions[Act_b]
    if BINOMPVALUE(Cnt_a, Cnt_a + Cnt_b, 0.5) ≤ α then
        return Act_a
    else
        return DETERMINEACTION(s, AS)
```

Algorithm 1 depicts the whole of determining actions, where **s** denotes a DMU. We first sample t concrete states in the DMU and input them to the trained DNN to obtain actions that will be stored in the *Actions* array. For continuous action spaces, we directly take the average of actions. Otherwise, we choose the first two actions with the highest occurrences. Following the practice in [6], the BINOMPVALUE function returns the p-value of the hypothesis test, where $Cnt_a \sim \text{Binomial}(Cnt_a + Cnt_b, 0.5)$. If the p-value is less than or equal to the statistical significance α, Act_a is the action for the corresponding DMU. Namely, if Cnt_a is much higher than Cnt_b, Act_a will be returned. Otherwise, we will repeat the action determination process.

2.4. Hyperparameter Setting

Algorithm 1 requires three hyperparameters: sampling count t, statistical significance α, and discretization vector D. Usually, we set t to 5, which experimentally shows a good balance between sampling time cost and final policy performance. Besides, α is used to calibrate the resampling threshold. Its setting value is related to t. We set it to 0.2 so that only when Act_a appears 5 or 4 times can it be regarded as the action of the DMU. The vector D determines the granularity of generated DMUs. We can adjust it according to the performance of the extracted policy. For instance, if the extracted policy performs worse than the benchmark DNN, we can decrease it to generate preciser DMUs. Otherwise, we can increase it appropriately to reduce the number of DMUs, which can reduce the verification time to some extent.

3. Model Checking with Extracted Policies

The DMUs of extracted policies are finite and their union covers the entire state space. They can be treated as states of the transition system. Therefore, the system driven by the extracted policy can first be transformed into a DMU-based transition system (TS). Then existing model checking techniques can be leveraged to verify its LTL properties [3].

Accordingly, we introduce the abstract and refinement techniques to solve the problems encountered in constructing transition systems. Then we combine these two operations into an algorithm to construct TS automatically. At last, we discuss the process of LTL model checking based on the constructed TS.

3.1. Abstraction and Refinement

3.1.1 Abstraction in Building Transition Relations

Since the concrete states contained in the DMU adopt the same action, as demonstrated in Figure 1, we can treat them as a whole to carry out a state transition based on system dynamics. However, the generated area (pink part) may be irregular due to the nonlinearity of system dynamics, so it is hard to formally represent the reachable area, which brings difficulties to the construction of transition relations between DMUs.

Therefore, our goal is to abstract the generated region into the form we can represent. We properly expand the irregular area to fit multiple DMUs exactly. Formally, let min_i and max_i represent the minimum and maximum of the irregular area on the i-th dimension, respectively. We use $[l_1, u_1) \times \ldots \times [l_n, u_n)$ to define the corresponding expanded area. Then for each dimension, we can uniquely determine l_i and u_i based on the following constraints, where R_i is mentioned in Section 2.2:

$$l_i \leq min_i,\ min_i < l_i + d_i,\ l_i \in R_i$$
$$max_i < u_i,\ u_i - d_i \leq max_i,\ d_i | (u_i - l_i).$$

Let $count_i = (u_i - l_i)/d_i$. Obviously, the expanded area contains $\Pi_{i=1}^{n} count_i$ DMUs. In Figure 1, four successor DMUs intersect the irregular area. As for the subsequent transition, we can decompose it into the transition of each DMU that constitutes the expanded area, thus forming an iterative construction procedure.

3.1.2 Property-Based Refinement of DMUs

The LTL property consists of atomic propositions. Therefore, it is necessary to determine the atomic propositions each state satisfies in the transition system. Otherwise, LTL properties cannot be verified. However, treating DMUs as states in the transition system may lead to ambiguity since concrete states in the same DMU cannot be guaranteed to all satisfy certain atomic proposition. For example, for the atomic proposition b in Figure 2, partial concrete states in the DMU satisfy b, and the others satisfy $\neg b$, which will

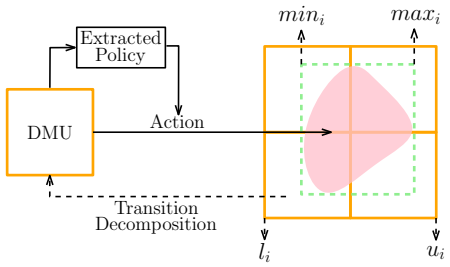


Figure 1: Computing direct successors of the DMU in a 2-dimensional control system.

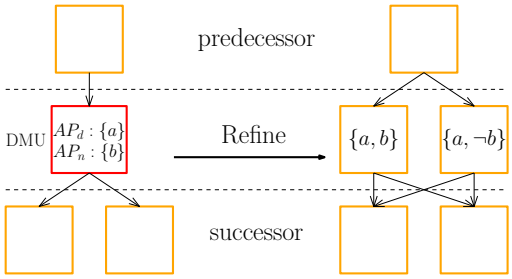


Figure 2: Property-based refinement of the DMU, where the red bounding box represents a nondeterministic DMU.

affect the verification of LTL properties like $F[\ b\ ]$. We call such DMUs and corresponding atomic propositions nondeterministic.

We refine the nondeterministic DMUs in the transition system constructed in Section 3.1.1 to eliminate ambiguity. In practice, we replace the nondeterministic DMU with multiple DMUs. Except for satisfying atomic propositions, the other properties of these substitute DMUs are the same as the original one, including the area, direct predecessors, and direct successors.

Formally, let AP_d define the set of propositions satisfied by the original DMU and $AP_n = \{\varphi_1, \ldots, \varphi_m\}$ represent the set of nondeterministic propositions. In Figure 2, AP_d and AP_n are $\{a\}$ and $\{b\}$, respectively. Then we define the set $AP_{sub} = \{\varphi_1, \neg\varphi_1\} \times \ldots \times \{\varphi_m, \neg\varphi_m\}$. Accordingly, there will be 2^m substitute DMUs. The i-th one satisfies the atomic propositions $AP_d \cup ap_i$, where $ap_i \in AP_{sub}$. If AP_n is empty, there will be only one substitute DMU that is exactly the original one.

3.2. Construction of Transition Systems

We combine the above two operations into an algorithm to construct the transition system, where the primary workflow is described in Algorithm 2.

We use $\mathbf{s}^0$ to denote the initial DMU and assume it is deterministic. For each DMU $\mathbf{s}$ fetched from *Queue*, we first determine its action based on Algorithm 1. Then we sequentially calculate the extreme values in each dimension of the irregular area generated by the state transition, i.e., min_i and max_i in Line 7. The set $\{\mathbf{s}^1, \ldots, \mathbf{s}^m\}$ represents the

Algorithm 2 Abstraction-Based TS Construction

Input: initial DMU $\mathbf{s}^0$, action space AS, state transition function f.
Output: transition system TS.

$Queue \leftarrow \{\mathbf{s}^0\}$
TS.setInitialState($\mathbf{s}^0$)
while $Queue \neq \emptyset$ **do**
  Fetch $\mathbf{s}$ from $Queue$
  $action \leftarrow$ DETERMINEACTION($\mathbf{s}$, AS)
  for $i = 1, \ldots, n$ **do**
    $[min_i, max_i] \leftarrow$ CALEXTREMUM(f, $\mathbf{s}$, $action$, i)
  $\{\mathbf{s}^1, \ldots, \mathbf{s}^m\} \leftarrow$ ABSTRACT($[min_1, max_1] \times \ldots \times [min_n, max_n]$)
  for $i = 1, \ldots, m$ **do**
    $\{\mathbf{s}^i_1, \ldots, \mathbf{s}^i_k\} \leftarrow$ REFINE($\mathbf{s}^i$)
    for $j = 1, \ldots, k$ **do**
      TS.addEdge($\mathbf{s} \rightarrow \mathbf{s}^i_j$)
      if $\mathbf{s}^i$ has not been traversed **then**
        Push $\mathbf{s}^i_j$ into $Queue$
return TS

DMUs that constitute the expanded area. For each DMU $\mathbf{s}^i$ that is transitioned from $\mathbf{s}$, we replace it with multiple deterministic states and add the edge from $\mathbf{s}$ to the substitute DMUs into the transition system. Finally, if DMU $\mathbf{s}^i$ has not been traversed, we will push all its substitute DMUs into *Queue* to be visited, thus ensuring that they have the same direct successors.

3.3. LTL Model Checking

The verification algorithm for LTL properties first constructs the product of the automata that is equivalent to the negative form of the property and TS, then checks whether there exists a path that can be accepted [3]. Namely, TS is proved to satisfy the property if no violated path is found.

Since we preserve all original paths of the system and solve problems by adding the reachable range, we can ensure the soundness of verification results. Soundness means that the LTL properties verified on the constructed TS are also true in the original control system.

Therefore, all that remains is to leverage existing model checking tools after expressing the constructed TS in the specified rules. We utilize *Spot* [7] to complete the subsequent verification work in practice.

4 Experiments and Evaluation

We intend to demonstrate three aspects of our approach through experiments: (i) performance comparison with the benchmark DNN, (ii) LTL verification results of four systems after TS construction, and (iii) method effectiveness compared to relevant works.

All experiments are performed on a workstation running Ubuntu 18.04 with a 32-core AMD CPU. We select two control systems from Gym [5], plus Tora [12] and 4-Car Platoon [23], as test systems. We introduce them below:

Mountain Car (MC) A car is positioned on a one-dimensional track between two hills. It is expected to drive up the right mountain where the position is 0.5.

Pendulum (PD) A pole can rotate around a fixed endpoint, where it is expected to swing up and remain upright.

Tora A cart fixed on the wall with springs can move freely on a frictionless surface. The arm on the cart can rotate freely around an axis. The controller is expected to stabilize the system to an equilibrium state.

4-Car Platoon (4CP) Four cars are supposed to drive in a platoon behind each other. An intuitive requirement is that the four cars cannot collide.

4.1 Performance Comparison

We utilize a framework [21] to train benchmark neural networks. Note that the action space is discrete in MC and continuous in other systems, so we use DQN [15] to train DNNs in MC and DDPG [18] to train DNNs in the others. We use the default settings in the framework for all other training hyperparameters, such as learning rate. Besides, the discretization vector D set for policy extraction in each system is listed in Table 1 (Column: Initial DMU).

We compare the performance of extracted policies and original DNNs via the episode reward value. We test 500 episodes for both policies. As shown in Figure 3, we use boxplots to depict the distribution of test results, where the black dots are outliers.

We can see that the metrics describing the distribution of episodic rewards, such as minimum and median, are close. Significantly, the performance of extracted policies in MC and Tora is marginally better than the DNNs. Therefore, we can conclude that the policy extraction method is reasonable and practical for these control systems.

4.2. LTL Verification Analysis

The LTL verification results of four systems driven by extracted policies are listed in Table 1.

MC We set constraints for MC. Let $p(s)$ and $v(s)$ be the position and velocity of the car at state s, respectively. Property $G[\ p(s) = 0.2 \rightarrow v(s) > 0.01\ ]$ states that the car's speed must be greater than 0.01 at position 0.2. The other property is that the car's position will eventually be greater than 0.5, formulated by $F[\ p(s) > 0.5\ ]$.

Both properties can be verified under the initial state space $[-0.5, -0.4999) \times [0, 10^{-4})$. Since the TS is the same in both test cases, the verification times are also close.

PD We also set two LTL properties for PD. Firstly, we set $G[\ |\theta(s)| < \frac{\pi}{2}\ ]$, where $\theta(s)$ and $\omega(s)$ denote the angle and angular velocity of the pole, respectively. The formula describes that the pole's angle must always be in $(-\frac{\pi}{2}, \frac{\pi}{2})$. Besides, we expect that the angular velocity will eventually

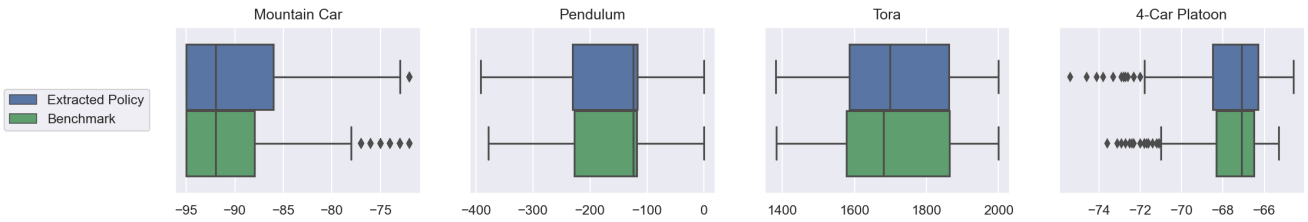


Figure 3: Performance comparison between the extracted policies and the DNNs (horizontal axis: the episode reward).

Table 1: LTL verification results of four systems driven by extracted policies.

Case	Initial DMU	LTL Property	Number	All	Verified	Time(s)
MC	$< [-0.5, 0.0], [10^{-4}, 10^{-4}] >$	$G[\ p(s) = 0.2 \rightarrow v(s) > 0.01\ ]$	1.2×10^6	✓	✓	2687
		$F[\ p(s) > 0.5\ ]$	1.2×10^6	✓	✓	2691
PD	$< [0, 0], [10^{-2}, 10^{-2}] >$	$G[\ \lvert\theta(s)\rvert < \frac{\pi}{2}\ ]$	728	✓	✓	5
		$G[\ \theta(s) < 0 \rightarrow F[\ \omega(s) > 0\ ]\ ]$	728	✓	✗	6
Tora	$< [0, 0, 0, 0],$ $[10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}] >$	$G[\ \lvert s_1\rvert < 2 \wedge \lvert s_2\rvert < 2$ $\wedge \lvert s_3\rvert < 2 \wedge \lvert s_4\rvert < 2\ ]$	1.0×10^6	✗	✓	2731
4CP	$< [0, 0, 0, 0, 0, 0, 0],$ $[10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}] >$	$G[\ d_1(s) > 0 \wedge$ $d_2(s) > 0 \wedge d_3(s) > 0\ ]$	9721	✓	✓	154

Remark: the **Number** of DMUs in TS; whether TS contains **All** reachable DMUs; whether the property is **Verified**, and the verification **Time**.

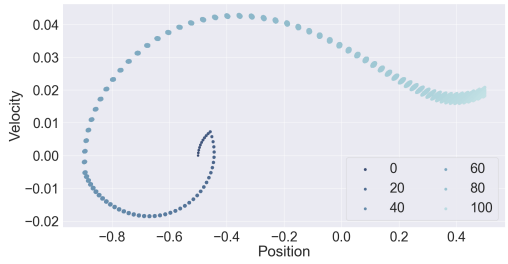


Figure 4: Traversal of DMUs over time steps in MC.

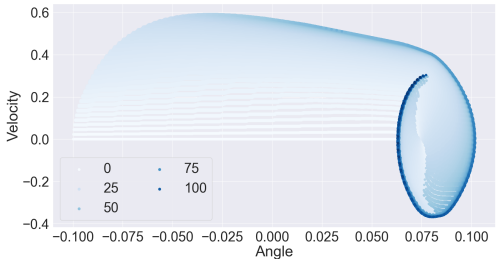


Figure 5: Traversal of DMUs over time steps in PD.

be greater than 0 if the angle is less than 0, i.e., $G[\ \theta(s) < 0 \rightarrow F[\ \omega(s) > 0\ ]\ ]$.

The second property fails to be verified because abstract methods applied in building transition relations add paths that violate the constraint.

Tora The equilibrium state for Tora is $[0, 0, 0, 0]$, so we expect the system can stay within $(-2, 2)^4$, which can be represented by $G[\ |s_1| < 2 \wedge |s_2| < 2 \wedge |s_3| < 2 \wedge |s_4| < 2\ ]$.

We limit the number of traversed DMUs to 10^6 to control the verification time. Therefore, the expected property can only be guaranteed within 24 time steps.

4CP We use $d_i(s)$ to denote the distance between the i-th car and $(i+1)$-th car, the constraint can be formulated as $G[\ d_1(s) > 0 \wedge d_2(s) > 0 \wedge d_3(s) > 0\ ]$.

Although 4CP is a 7-dimensional system, reachable DMUs are limited. The property can be verified among around 10^4 DMUs.

Evaluation Since the time required to build the transition system is much more than the execution time of *Spot*, the verification time is proportional to the number of traversed DMUs. However, compared with the state space, reachable DMUs are limited. For example, taking the center concrete state in the DMU, we depict the traversal of DMUs over time steps in MC and PD in Figure 4 and Figure 5, respectively, where different colors indicate that DMUs are traversed at different time steps. It can be seen that both systems operate in a fraction of the state space, which can make our method immune to the state explosion problem. Therefore, the combination of policy extraction and TS construction can ensure efficient verification of systems with limited reachable ranges.

Table 2: Performance comparison with ReachNN*.

Case	Neural Network	Ours	ReachNN*
B1	Tanh(2×20)	**389**	Unknown
	Tanh(2×100)	**401**	Unknown
B2	Sigmoid(2×20)	91	**22**
	Sigmoid(2×100)	**96**	105
MC	Sigmoid(2×16)	**2567**	Unknown
	Sigmoid(2×200)	**2574**	Unknown

4.3 Method Comparison

To our best knowledge, few related methods can verify properties other than *safety* and *liveness*, such as LTL in this paper. Therefore, we only compare the verification of *liveness* properties with ReachNN* [9].

For simplicity, we follow most of the experimental settings in [11], including test systems and verification properties. Therefore, we only list the necessary parameters in Table 2. There are two types of verification results: verification passes (verification time) or fails (Unknown). The discretization vector values in B1 and B2 are $[10^{-3}, 10^{-3}]$ and $[10^{-3}, 10^{-4}]$, respectively.

Our method outperforms ReachNN* in most cases. More importantly, the scalability of reachability analysis methods on which ReachNN* is based is limited by the structure and scale of DNNs. However, the trained DNN is a black box to our method via policy extraction, so the verification time is independent of DNNs.

5 Related Work

There is a growing literature on formal verification of DRL systems. Reachability analysis methods can calculate reachable sets of systems at each time step. For instance, Fan *et al.* approximated the DNN controller with Bernstein polynomials [9] while Ivanov *et al.* proposed a Taylor-model-based reachability algorithm to improve the scalability [11]. Besides, shielding can prevent DRL systems from exhibiting unsafe behavior [1, 23]. However, these methods can only verify the safety and liveness properties of DRL systems, while our approach can verify more complex temporal properties to guarantee applicability.

Works made by [8, 14] aim to model checking the DRL-driven systems via techniques on formal verification of DNNs. However, the scalability of their methods is limited by the size of neural networks, so the length of counterexamples obtained in experiments is limited. In contrast, the scalability of our method is DNN-independent.

6 Conclusion and Future Work

We proposed an LTL model checking approach to DRL verification. Our approach relies on extracting interpretable policies from trained DNNs and modeling DRL systems as a finite-state transition system using the extracted policies. It supports model checking of more complex temporal properties of DRL systems except for safety properties. We applied it to the DRL systems trained for four classic control problems. The experimental results show the effectiveness of our approach.

Our approach demonstrated the feasibility of extracting interpretable policies to substitute for inexplicable neural networks for formal verification. We plan to apply our approach to more complex DRL systems and devise more efficient model checking algorithms to improve scalability.

Acknowledgement

This work is supported by National Key Research Program (2020AAA0107800), Shanghai Science and Technology Commission (20DZ1100300), Shanghai Artificial Intelligence Innovation and Development Fund (2020-RGZN-02026), Shenzhen Institute of Artificial Intelligence and Robotics for Society (AC01202005020), NSFC projects (61872146, 62161146001). Yang Wang and Min Zhang are the corresponding authors.

References

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, et al. Safe reinforcement learning via shielding. In *AAAI*, 2018.

[2] Edoardo Bacci and David Parker. Probabilistic guarantees for safe deep reinforcement learning. In *FORMATS*, pages 231–248, 2020.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[4] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *NIPS*, 31, 2018.

[5] Greg Brockman et al. OpenAI Gym, 2016.

[6] Jeremy Cohen et al. Certified adversarial robustness via randomized smoothing. In *ICML*, pages 1310–1320, 2019.

[7] Duret-Lutz et al. Spot 2.0—a framework for ltl and ω-automata manipulation. In *ATVA*. Springer, 2016.

[8] Tomer Eliyahu, Yafim Kazak, Guy Katz, and Michael Schapira. Verifying learning-augmented systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 305–318, 2021.

[9] Jiameng Fan et al. Reachnn*: A tool for reachability analysis of neural-network controlled systems. In *ATVA*, pages 537–542, 2020.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Radoslav Ivanov et al. Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In *International Conference on Computer Aided Verification*, 2021.

[12] Mrdjan Jankovic, Daniel Fontaine, and Petar V KokotoviC. Tora example: cascade-and passivity-based control designs. *IEEE Transactions on Control Systems Technology*, 4(3), 1996.

[13] Nathan Jay et al. Internet congestion control via deep reinforcement learning. *CoRR*, abs/1810.03259, 2018.

[14] Yafim Kazak, Clark Barrett, Guy Katz, and Michael Schapira. Verifying deep-rl-driven systems. In *Proceedings of the 2019 Workshop on Network Meets AI & ML*, pages 83–89, 2019.

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, et al. Continuous control with deep reinforcement learning. In *ICLR'16*, 2016.

[16] Yen-Chen Lin et al. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[17] Björn Lütjens, Michael Everett, and Jonathan P How. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, 2019.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[20] Pierre El Mqirmi, Francesco Belardinelli, and Borja G León. An abstraction-based method to check multi-agent deep reinforcement-learning behaviors. *arXiv preprint arXiv:2102.01434*, 2021.

[21] Kei Ota. TF2RL. https://github.com/keiohta/tf2rl/, 2020.

[22] Christian Szegedy et al. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[23] He Zhu et al. An inductive synthesis framework for verifiable reinforcement learning. In *PLDI*, pages 686–701, 2019.

Formal Verification of COCO Database Framework Using CSP

Peimu Li, Jiaqi Yin, Huibiao Zhu*
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China

***Abstract*—Nowadays, many applications are built on distributed databases for scalability and high availability. Therefore, the architecture design of distributed databases needs to satisfy some functional properties to ensure that the database can perform transactions reliably and efficiently. COCO is a distributed OLTP database that supports epoch-based commit and replication and two variants of optimistic concurrency control which use physical time or logical time. In this paper, we first use process algebra CSP to model COCO's architecture. Then we use model checker PAT to verify seven properties, including deadlockfree, consistency, availability, partition tolerance (CAP), and basically availability, soft state, eventual consistency (BASE). The results show COCO's commit and replication protocol satisfy the CAP theorem, and two optimistic concurrency control variants satisfy the BASE theorem.**

***Index Terms*—Distributed OLTP Database, Process Algebra, COCO, Modeling, Verification**

I. Introduction

Many distributed OLTP databases use a shared-nothing architecture for scale out and data partitioning to achieve the scalability of data storage [1], [2]. When the data required by a transaction belongs to multiple data partitions, the system needs a protocol that can coordinate the collaborative work of multiple partitions to ensure the normal execution of the transaction, so a two-phase commit protocol (2PC) is proposed [3]. It is well known that 2PC causes significant performance degradation in distributed databases [4]. There have been some improvements aimed at the defects of 2PC, but most of them require some assumptions which are difficult to achieve, e.g., read/write sets of each transaction has to be known before execution [5].

Lu et al. [6] proposed epoch-based commit and replication, which is an improved protocol based on 2PC, and implemented it in distributed database COCO. The COCO database also supports two variants of optimistic concurrency control: physical time and logical time OCC, which can serialize transactions in physical or logical time [7].

The design of a distributed database architecture often needs to satisfy many functional properties. Fox et al. [8] put forward the CAP theory based on the characteristics of distributed systems. This theory explains three properties that restrict each other in the design of distributed system architecture: consistency, availability, and partition tolerance. Any framework can only satisfy two of them, but not all of them. Pritchett [9] proposed BASE theory, which is a compromise solution for the design of distributed systems against limitations of CAP theory. It allows data inconsistency for a period of time and satisfies three properties: basic availability, soft state and eventually consistency.

At present, the mainstream method of detecting the performance of the database are benchmark testing and load testing. Benchmark testing refers to a test method that quantitatively compares certain specific performance indicators in the system. Currently, the popular benchmarking tools include Facebook's LinkBench [10], Yahoo's YCSB [11] and BigDataBench [12]. Load testing is to test performance of the tested object by continuously increasing the task volume to the tested object until a certain index reaches or exceeds the expectation or a certain resource is exhausted.

Actually, for the reason that the test workload and benchmarks are artificially set, and the test results are directly affected by the hardware performance, the test results still can be improved. In order to solve these challenges, this paper applies a formal method called CSP to verify properties of the database architecture. CSP [13] is an algebra theory proposed by C. A. R. Hoare. It is an abstract language designed to describe process communication in concurrent systems. We use CSP to abstract the architecture and use the model checker PAT [14] to check the properties of the model. In this way, we can reduce the impact of hardware on the verification process and ensure completeness.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the epoch-based commit and replication, two optimistic concurrency controls in COCO and process algebra CSP. In Section III, we use CSP to model the commit protocol and two types of concurrency control in COCO. In Section IV, we implement the achieved formed model of Section III in PAT, and give the definition of the properties that need to be verified and the verification results. Section V concludes the paper and provides some future work.

II. Background

In this section, we introduce the overall architecture of COCO database and process algebra CSP.

A. Epoch-based Commit and Replication

Epoch-based commit and replication contains two protocols: (1) a commit protocol and (2) a replication protocol.

The commit protocol is used to commit completed transactions at the end of the current epoch, shown in Fig.

*Corresponding author: hbzhu@sei.ecnu.edu.cn (H. Zhu).

DOI reference number: 10.18293/SEKE2022-072

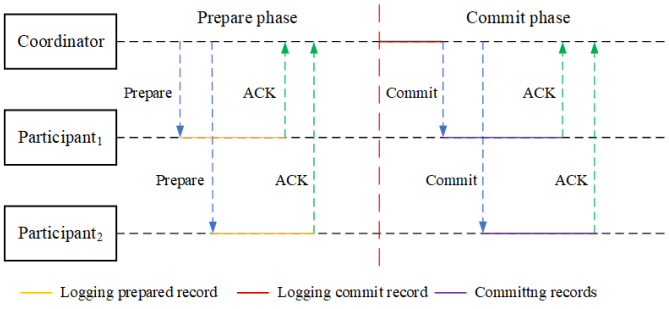


Fig. 1. Epoch-based commit

1. Epoch-based commit contains a $prepare$ phase and a $commit$ phase. In the $prepare$ phase, the coordinator node first sends a preparation message to all other participant nodes. When a participant node receives the preparation message, it prepares to commit all transactions in the current epoch by logging a durable prepared write record with all the transaction IDs (TIDs) of ready-to-commit transactions as well as the current epoch number. When a participant node logs all the necessary write records, it then replies an acknowledgement to the coordinator.

If any participant node fails to send an acknowledgement message due to failures, all transactions in the current epoch must be aborted. Otherwise the coordinator writes a durable commit record with the global current epoch number, and increases the global current epoch number. Then the coordinator sends a commit message to all participant nodes. When a participant node receives a commit message, it commits all the ready-to-commit transactions in the last epoch. Note that even if some transactions in the epoch are aborted due to conflicts, they do not affect the commits of other transactions in the epoch. At this point, the writes of all ready-to-commit transactions in the last epoch are visible to all other users. In the end, all participants send acknowledgement messages to the coordinator, and then start to execute the transactions of the next epoch.

The replication protocol is needed to guarantee consistency and availability of the database. The most common approach is the primary-backup replication. In COCO, atomicity and durability are satisfied at the end of the epoch, so that COCO can perform the replication on backup databases asynchronously. Therefore, after the data update on the primary is completed, the primary can release all locks.

B. Physical Time and Logical Time OCC

Physical Time OCC (PT-OCC) and Logical Time OCC (LT-OCC) are two distributed variants of OCC. In PT-OCC, the transaction needs to lock the data items in the write set. A lock request is only sent to the primary replica of each record. If the data item has been locked by other transactions, the transaction is directly aborted. If the record is in the write and read sets of the transaction at the same time, the transaction will verify whether the TID of the corresponding record in the read set is consistent with that in the primary. If the TID is inconsistent, it means that the record was updated by other transactions during the execution phase, so the transaction simply aborts.

When the transaction has locked the write set, it begins to verify its read set. A read verification request is only sent to the primary replica of each record. If a record in the primary is locked by other transactions or the TID is inconsistent with the record in the read set, then the transaction simply aborts. At the same time, COCO generates a new TID for the transaction. TID marks the order between transactions.

After successfully verifying the read set, the transaction writes the result back to the database. A write request is sent to the primary replica of each record. After the result is written, the lock of the record in the primary replica is released immediately, and then the result is written to other replicas asynchronously.

In LT-OCC, a serialized transaction can read, write and commit in the space of logical time. Each record in the database is associated with two logical timestamps $[wts, rts]$. Here, wts is the timestamp of the last modification time of this record. rts represents the effective time of this record, which means that it is valid to read this record at any logical time ts that satisfies $wts \leq ts \leq rts$. The control flow of the LT-OCC algorithm is basically the same as that of PT-OCC. The difference is TID is used to identify transactions in PT-OCC and $[wts, rts]$ is used in LT-OCC.

C. CSP

CSP is a formal language which has an important impact on the development of Golang. The following is some widely used CSP syntax:

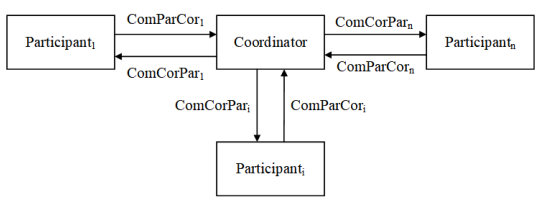
- $SKIP$ denotes that a process terminates successfully.
- $a \rightarrow P$ indicates the process executes action a first, and then behaves like P.
- $P \square Q$ represents the general choice. It behaves like P or Q, and the environment decides the selection.
- $P||Q$ stands for concurrent execution of P and Q. The shared actions must be executed synchronously.
- $P|||Q$ denotes that P interleaves Q.
- $c?x \rightarrow P$ indicates that a value is recieved through channel c and process assigns it to variable x, and then behaves like P.
- $c!e \rightarrow P$ denotes the process sends the value of expression e through channel c first, and then behaves like P.
- $P;Q$ represents sequential execution, process executes P, then executes Q after P which first terminates.
- $P \triangleleft b \triangleright Q$ denotes the condition. If b is true, then process behaves like P. Otherwise process behaves like Q.

III. Modeling COCO Database Framework

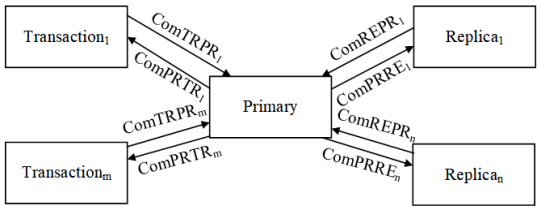
In this section, we use process algebra CSP to model COCO architecture. First, we introduce the messages and channels used in our model, and then we respectively introduce the CSP models of COCO architecture. COCO architecture includes epoch-based commit protocol, replication protocol and two variants of optimistic concurrency control.

A. Messages and Channels

Before modeling the commit protocol and concurrency control algorithm in COCO, we need to define the messages, and channels used for modeling. We define multiple channels for data interaction between various modules in the COCO database system. Fig. 2 gives the channels of communication in COCO:



(a) Channels of Epoch-based Commit



(b) Channels of Optimistic Concurrency Control and Replication

Fig. 2. Channels of COCO database

In order to define the message conveniently, we define three message content sets: REQ, ACK and $DATA$, which represent content of the request, confirmation and data messages respectively. Based on the above definitions, we design multiple types of messages for information exchange between entities. These messages are defined as follows:

$MSG =_{df} MSG_{req} \cup MSG_{ack} \cup MSG_{data}$

$MSG_{req} =_{df} \{msg_{req}.C.P.Content \mid msg_{req} \in TYPE , C \in Coordinator, P \in Participant, Content \in REQ, \}$

$MSG_{ack} =_{df} \{msg_{ack}.P.C.Content \mid c \in Coordinator , P \in Participant, Content \in ACK, \}$

$MSG_{data} =_{df} \{msg_{data}.P.R.Content \mid P \in Primary, R \in Replica, Content \in DATA, \}$.

In COCO framework, we define MSG_{req} to represent the request messages, including six types of requests: prepare, commit, abort, read, write, and lock. Six types of requests are in $TYPE$ set. MSG_{ack} represents confirmation of the request message, and MSG_{data} represents data information passed by the transaction in the read and write process.

B. Epoch-based Commit and Replication Modeling

The epoch-based commit protocol includes two types of processes: coordinator and participants. The coordinator process is modeled as follows:

$$
\begin{aligned}
&Coordinator() =_{df} (|||i : \{1..N\}@ComCorPar[i]!msg_{prep}.\\
&\quad C.P.epoch\_num \rightarrow Skip); (|||i : \{1..N\}@ComParCor\\
&\quad [i]?msg_{ack}.P.C.R\_C\_TID \rightarrow Check\{if(msg_{ack} ==\\
&\quad No)\{hasNo = true\}\} \rightarrow Skip); decide\{if(hasNo ==\\
&\quad true)\{choice = ABORT\}else\{choice = COMMIT\}\}\\
&\quad \rightarrow CoordCommitPhase(choice)
\end{aligned}
$$

$$
\begin{aligned}
&CoordCommitPhase(choice) =_{df}\\
&\quad if(choice == COMMIT)\{\\
&\qquad writeCommitRecord \rightarrow increaseEpoch\{epoch\_n\\
&\qquad um = epoch\_num + 1\} \rightarrow Skip;\\
&\qquad (|||i : \{1..N\}@ComCorPar[i]!msg_{commit}.C.P.epo\\
&\qquad ch\_num - 1 \rightarrow Skip); (|||i : \{1..N\}@ComParCor\\
&\qquad [i]?msg_{ack}.P.C.com\_TID \rightarrow Skip);\\
&\qquad releaseResults \rightarrow Coordinator()\\
&\quad \}\\
&\quad else\{\\
&\qquad (|||i : \{1..N\}@ComCorPar[i]!msg_{abort}.C.P.epo\\
&\qquad ch\_num \rightarrow Skip);\\
&\qquad (|||i : \{1..N\}@ComParCor[i]?msg_{ack}.P.C.abor\\
&\qquad t\_TID \rightarrow Skip);\\
&\qquad abort \rightarrow Coordinator()\\
&\quad \}
\end{aligned}
$$

In the $prepare$ phase, the $coordinator$ first sends a MSG_{prep} to N numbers of $participants$, and then receives MSG_{ack} sent by $participant$ nodes. If there are $participant$ nodes that cannot commit the transaction, the $choice$ is assigned to $ABORT$, otherwise it is assigned to $COMMIT$. In the $commit$ phase, if $choice == ABORT$, then the $coordinator$ needs to send MSG_{abort} to all $participant$ nodes to abort all transactions in this epoch. After receiving MSG_{ack} from all $participant$ nodes, it re-executes all transactions in the current epoch. If $choice == COMMIT$, $coordinator$ writes a durable commit record with $epoch\_num$, and then increases $epoch\_num$. It also sends MSG_{commit} to all $participant$ nodes, and then receives MSG_{ack} sent by $participant$ nodes. After completing above tasks, the $coordinator$ releases results to users and then executes the commit of the next epoch.

The definition of the participant process is as follows:

$$
\begin{aligned}
&Participant(i) =_{df} ComParCor[i]?msg_{prep}.C.P.epoch\_num\\
&\rightarrow \begin{pmatrix} logWriteRecord \rightarrow ComParCor[i]!Yes.\\ P.C.R\_C\_TID \rightarrow ParComPhase(i)\\ \Box\\ ComParCor[i]!No.P.C.R\_C\_TID\\ \rightarrow ParComPhase(i) \end{pmatrix}\\
&ParComPhase(i) =_{df} ComCorPar[i]?msg_{req}.C.P.epoch\_num\\
&\rightarrow \begin{pmatrix} ComParCor[i]!msg_{ack}.P.C.com\_TID\\ \rightarrow Participant(i)\\ \triangleleft(msg_{req} == msg_{commit})\triangleright\\ ComParCor[i]!msg_{ack}.P.C.abort\_TID\\ \rightarrow Participant(i) \end{pmatrix}
\end{aligned}
$$

In $prepare$ phase, the $participant$ first receives the MSG_{prep} sent by the $coordinator$, then if the $participant$ successfully writes the commit record, a Yes MSG_{ack} is sent directly to the $coordinator$. Otherwise, the $participant$ sends No MSG_{ack}. In $commit$ phase, the $participant$ first waits to receive MSG_{req} from the $coordinator$. If $MSG_{req} == MSG_{commit}$, then the $participant$ commits the result and sends a commit message to the $coordinator$. If $MSG_{req} == MSG_{abort}$, the $participant$ sends an abort

message. After that, $participants$ and the $coordinator$ synchronize to enter the next epoch of transaction commit.

Based on the above modeling, an overall epoch-based commit protocol is defined as follows:

$$Epoch\_commit() =_{df} Coordinator()||(|||i:\{1...N\}@Participant(i))$$

The epoch-based commit protocol consists of a $coordinator$ and several $participants$. The $coordinator$ and each $participant$ execute concurrently, and freely interleave execution between $participant$ processes.

The replication protocol mainly includes two types of processes, $Primary\_replication$ and $Replica$. The $Primary\_replication$ is responsible for controlling and coordinating entire replication execution on the primary database. $Replica$ is a process that runs on the replica server and cooperates with the primary. The definitions are as follows:

$$\begin{aligned}
&Primary_replication(records) =_{df} (|||i:\{1..N\}@ComP\\
&\quad RRE[i]!msg_{write}.P.R.records \rightarrow ComREPR[i]?ms\\
&\quad g_{ack}.R.P.content \rightarrow Primary_replication(records))\\
&Replica(i) =_{df} ComPRRE[i]?msg_{write}.P.R.records \rightarrow\\
&\quad WriteBack(i,records) \rightarrow ComREPR[i]!\\
&\quad msg_{ack}.R.P.Yes \rightarrow Replica(i)\\
&WriteBack(i,records) =_{df}\\
&\left(\begin{array}{c} Skip \\ \lhd(records == null)\rhd \\ \left(\begin{array}{c} write\{replica_i.record = record\} \rightarrow \\ WriteBack(i,records') \end{array}\right) \\ \lhd(replica_i.record.tid < record.tid)\rhd \\ WriteBack(i,records') \end{array}\right)
\end{aligned}$$

The $WriteBack$ first determines whether the $records$ is empty. If it is empty, the process terminates directly. Otherwise, if the database record's tid is greater than the tid of $records$, it indicates that the record has been updated by other transactions executed later, so the replica directly skips this record. Otherwise, it updates the record, and then calls $WriteBack(i, records')$. $records'$ contains all $records$ elements except the first record.

$$\begin{aligned}
&Replication(records) =_{df}\\
&Primary_replication(records)||(|||i:\{1...N\}@Replica(i))
\end{aligned}$$

The overall protocol consists of a $primary\_replication$ and N $replica$ processes. The modeling is shown above.

C. PT-OCC and LT-OCC Modeling

OCC can be roughly divided into three phases: (1) locking the write set, (2) validating the read set, (3) writing back to the database. However, there is a difference between PT-OCC and LT-OCC in phase (2), and the other phases are basically the same. Based on the above facts, we can perform unified modeling on phases (1) and (3) of the two algorithms, and model the phase (2) independently.

In the first phase, the transaction sends a MSG_{lock} to the primary of the record, and then receives the MSG_{ack} sent by the primary. If $content == Yes$, then the transaction continues to send the request for the next record, otherwise the transaction releases locked records and aborts. The definition of $Trans\_lock$ and $Releaselock$ are as follows:

$$\begin{aligned}
&Trans_lock(i,write_set) =_{df} Skip \lhd (write_set == null)\rhd\\
&\left(\begin{array}{c} ComTSPR[i]!msg_{lock}.T.P.record \rightarrow ComPRTS[i]?\\ msg_{ack}.P.T.content \rightarrow \\ \left(\begin{array}{c} Trans_lock(i,write_set') \\ \lhd(content == Yes)\rhd \\ Releaselock(i,lockedRecords) \rightarrow Skip \end{array}\right) \end{array}\right)\\
&Releaselock(i,records) =_{df} Skip \lhd (records == null)\rhd\\
&\quad release\{database.record.locked = false\} \rightarrow\\
&\quad Releaselock(i,records')
\end{aligned}$$

The definition of $primary$ in the first phase is as follows:

$$\begin{aligned}
&Primary_lock() =_{df} |||i:\{1...N\}@ComTSPR[i]?msg_{lock}.\\
&\quad T.P.record \rightarrow find\{if(record \notin read_set)\{record.tid\\
&\quad = database.record.tid\}\} \rightarrow\\
&\left(\begin{array}{c} Locking\{database.record.locked = True\} \rightarrow \\ ComPRTS[i]!msg_{ack}.P.T.Yes \rightarrow Primary_lock() \\ \lhd(database.record.locked == false\wedge \\ record.tid == database.record.tid)\rhd \\ ComPRTS[i]!msg_{ack}.P.T.No \rightarrow Primary_lock() \end{array}\right)
\end{aligned}$$

If $record \notin read\_set$, $primary$ reads $database.record.tid$ as the tid of $record$. If the record is not occupied by others and $record.tid == database.record.tid$, the $primary$ can lock and send Yes to the transaction, otherwise send No.

Next, the algorithm verifies whether the $records$ in the read set are still valid. We use $Primary\_valid$ to describe the behavior of the primary in the validation phase, and its definition is as follows:

$$\begin{aligned}
&Primary_valid() =_{df} |||i:\{1...N\}@(ComTSPR[i]?\\
&\quad msg_{valid}.T.P.record \rightarrow ComPRTS[i]!msg_{data}.\\
&\quad P.T.database_record \rightarrow Primary_valid())
\end{aligned}$$

Transactions in the verification phase have different behaviors under different concurrency control. The verification phase of the transaction in PT-OCC and LT-OCC are defined as follows:

$$\begin{aligned}
&Trans_valid_PT(i,read_set) =_{df} Skip \lhd (read_set == null)\\
&\rhd\left(\begin{array}{c} ComTSPR[i]!msg_{valid}.T.P.record \rightarrow ComPRT\\ S[i]?msg_{data}.P.T.content \rightarrow \\ \left(\begin{array}{c} validate \rightarrow Trans_valid_PT(i,read_set') \\ \lhd(content.locked == false\wedge \\ record.tid == content.record.tid)\rhd \\ Releaselock(i,lockedRecords) \rightarrow Stop \end{array}\right) \end{array}\right)\\
&Trans_valid_LT(i,read_set) =_{df} Skip \lhd (read_set == null)\\
&\rhd\left(\begin{array}{c} ComTSPR[i]!msg_{valid}.T.P.record \rightarrow ComPRT\\ S[i]?msg_{data}.P.T.content \rightarrow \\ \left(\begin{array}{c} validate \rightarrow Trans_valid_LT(i,read_set') \\ \lhd(record.wts == content.wts\wedge \\ (content.rts \geq record.tid\vee \\ content.locked == false))\rhd \\ Releaselock(i,lockedRecords) \rightarrow Stop \end{array}\right) \end{array}\right)
\end{aligned}$$

The main difference between the transaction in LT-OCC and PT-OCC is the condition for judging whether the record is successfully verified. In LT-OCC, the wts of the $record$ in the $read\_set$ of the transaction must be the same as the wts of the corresponding record in the database. For the case where the rts of the $record$ is less than the tid of the transaction, as long as the record has not been locked by

other transactions, the transaction also considers the $record$ to be valid and the verification is successful.

After verifying all records in $read\_set$, the transaction writes the records in $write\_set$ into the $primary$ database. The transaction definition in the write phase is as follows:

$$Trans\_write(i, write\_set) =_{df} Skip \triangleleft (write\_set == null) \triangleright \begin{pmatrix} ComTSPR[i]!msg_{write}.T.P.record \rightarrow ComPRT \\ S[i]?msg_{ack}.P.T.content \rightarrow ComTSPR[i]! \\ msg_{release}.T.P.write\_set \rightarrow Trans\_write(\\ i, write\_set') \end{pmatrix}$$

The transaction first checks whether there are any records in $write\_set$. If not, the write is complete and the transaction can be terminated at this point. Otherwise, the transaction sends a write request MSG_{write} to the $primary$, and then receives the MSG_{ack} from the $primary$. After that, the transaction sends a request $MSG_{release}$ to release the record lock to the $primary$, and then tries to write the next record. The primary definition in write phase is as follows:

$$Primary\_write() =_{df} |||i : \{1...N\}@(ComTSPR[i]? msg_{write}.T.P.record \rightarrow write\{database.record = record\} \rightarrow ComPRTS[i]!msg_{ack}.P.T.Yes \rightarrow ComTSPR[i]?msg_{release}.T.P.write\_set \rightarrow Release(i, record) \rightarrow Primary\_replication(record))$$

The $primary$ receives the write request MSG_{write} from the transaction, and then writes the $record$. After successfully writing the $record$, the $primary$ sends a confirmation message MSG_{ack} to the transaction. After receiving the release request from the transaction, the $primary$ calls the $Release$ function to release the lock of the record, and then enters the replication phase.

The overall definition of concurrency control algorithm includes three parts: transaction, primary and replica. The definition of $Primary$ is as follows:

$$Primary() =_{df} Primary\_lock()|||Primary\_valid()||| Primary\_write()$$

The definition of $Transaction\_PT$ in PT-OCC is as follows. $Transaction\_LT$ in LT-OCC differs only in the second phase.

$$Transaction\_PT(i, read\_set_i, write\_set_i) =_{df} (Transaction\_lock(i, write\_set_i); Transaction\_valid\_PT(i, read\_set_i); Transaction\_write(i, write\_set_i))$$

The definitions of PT-OCC and LT-OCC are as follows:

$$PT\_OCC() =_{df} (|||i : \{1...N\}@Transaction\_PT(i, read\_set_i, write\_set_i))||Primary()||(|||i : \{1...N\}@Replica(i))$$

$$LT\_OCC() =_{df} (|||i : \{1...N\}@Transaction\_LT(i, read\_set_i, write\_set_i))||Primary()||(|||i : \{1...N\}@Replica(i))$$

Both $PT - OCC$ and $LT - OCC$ are composed of concurrent execution of multiple $transaction$, a $Primary$ and multiple $Replica$ processes.

IV. Implementation and Verification

In this section, based on the achieved formed model in Section III, now we conduct verification of the properties abstracted from the specification.

A. Properties

a) Deadlockfree: This property refers to the situation in which processes are never deadlocked. PAT provides atomic statements to verify deadlockfree.

b) Consistency: This property asserts during the execution of a transaction, data can only be converted from one consistency state to another consistency state.

$$\#define\ Consistency\ (\wedge i : \{1...N\}record_i == last\_record) \vee (\wedge i : \{1...N\}record_i == cur\_record)$$
$$\#assert\ Epoch\_commitl() \mid=\ Consistency$$

c) Availability: It means every request in a distributed system can be responded to.

$$\#define\ Availability\ (hasNo == True \wedge finished == True)$$
$$\#assert\ Epoch\_commit() \mid=\ Availability$$

d) Partition Tolerance: It means that when a node or network partition in a distributed system fails, the entire system can still provide external services that satisfy consistency and availability.

$$\#define\ PartitionTolerance\ finished == True$$
$$\#assert\ Epoch\_commit() \mid=\ PartitionTolerance$$

e) Basically Availability: This property means that when some requests failure or unpredictable failures occur in the system, the system can still guarantee the normal execution of most transactions.

$$\#define\ BasicallyAvailability\ (existCrash == True) \wedge (available == True)$$
$$\#assert\ PT\_OCC() \mid=\ BasicallyAvailability$$
$$\#assert\ LT\_OCC() \mid=\ BasicallyAvailability$$

f) Soft State: This property refers to allowing the data in the system to have an intermediate state, and this state does not affect the overall availability of the system.

$$\#define\ SoftState\ (\vee i : \{1...N\}record_i! = last\_record) \wedge (\vee i : \{1...N\}record_i! = cur\_record)$$
$$\#assert\ PT\_OCC() \mid=\ SoftState$$
$$\#assert\ LT\_OCC() \mid=\ SoftState$$

g) Eventually Consistency: It refers to the fact that all data copies in the system can finally reach a consistent state after a period of synchronization without the guarantee of strong consistency of system data.

$$\#define\ EventuallyConsistency\ EG((\wedge i : \{1...N\}record_i == last\_record) \vee (\wedge i : \{1...N\}record_i == cur\_record))$$

$$\#assert\ PT\_OCC()\ |=\ EventuallyConsistency$$
$$\#assert\ LT\_OCC()\ |=\ EventuallyConsistency$$

B. Results

We use the model checker PAT to verify the main frameworks of COCO distributed database such as epoch-based commit and replication, PT-OCC and LT-OCC. The verification results are shown in Fig. 3 and Fig. 4. The

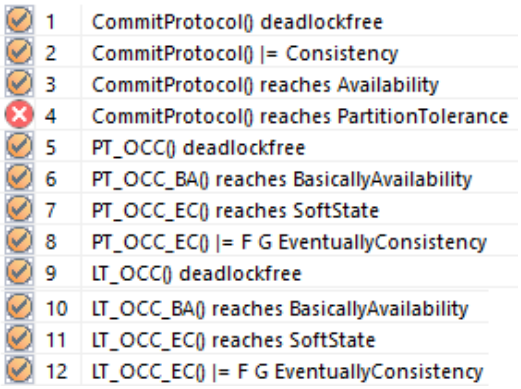


Fig. 3. The Verification Results of COCO Database Framework

Fig. 4. The Details of the Partial Verification Results

epoch-based commit protocol executed at the end of the epoch satisfies the consistency and availability but does not satisfy the partition tolerance(i.e. the 4th property in Fig. 3), which is in line with the CAP theory. During an epoch, the two types of concurrency control satisfy basically availability, soft state and eventually consistency, which meet the BASE theory. From the analysis of the above results, the COCO distributed database guarantees basic availability within an epoch and at the same time can satisfy strong consistency at the end of the epoch.

V. Conclusion and Future Work

COCO is a distributed database that regards the epoch as the unit of transaction commit and uses optimistic concurrency control. This paper used process algebra CSP to model COCO's epoch-based commit and replication, physical time OCC and logical time OCC, and implemented these models in the model checker PAT. The CAP and BASE theories put forward the properties that the distributed system architecture needs to satisfy, and we verified the properties of COCO in an epoch cycle. It has been verified that (1) epoch-based commit and replication satisfy consistency and availability but not partition tolerance, and (2) PT-OCC and LT-OCC satisfy basic availability, soft state, and eventually consistency. This shows that COCO can guarantee high availability during an epoch cycle, and can also guarantee consistency at the end of the epoch. In the future, we will verify the isolation of COCO and sequential consistency of concurrency control.

VI. Acknowledgements

This work was partly supported by the National Key Research and Development Program of China (Grant No. 2018YFB2101300), the National Natural Science Foundation of China (Grant Nos. 61872145, 62032024), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing (East China Normal University).

References

[1] Shute J, Vingralek R, Samwel B, et al. F1: A Distributed SQL Database That Scales. PVLDB, 2013, 6(11): 1068–1079.
[2] Verbitski A, Gupta A, Saha D, et al. Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases. SIGMOD, 2017: 1041–1052.
[3] Mohan C, Lindsay B, and Obermarck R. Transaction Management in the R* Distributed Database Management System. TODS, 1986, 11(4): 378–396.
[4] Bailis P, Fekete A, Franklin M, Ghodsi A, Hellerstein J, and Stoica I. Coordination Avoidance in Database Systems. PVLDB, 2014, 8(3): 185–196.
[5] Lin Q, Chang P, Chen G, Ooi B C, Tan K, and Wang Z. Towards a Non-2PC Transaction Management in Distributed Database Systems. SIGMOD, 2016: 1659–1674.
[6] Lu, et al. Epoch-based Commit and Replication in Distributed OLTP Databases. PVLDB, 2021, 14(5): 743-756.
[7] Kung H T and Robinson J T. On Optimistic Methods for Concurrency Control. TODS, 1981, 6(2): 213–226.
[8] Fox A, Brewer E A. Harvest, Yield, and Scalable Tolerant Systems. IEEE, 1999: 174-178.
[9] Pritchett D. BASE: An Acid Alternative: In Partitioned Databases, Trading some Consistency for Availability can Lead to Dramatic Improvements in Scalability. Queue, 2008, 6(3): 48-55.
[10] Timothy A, et al. LinkBench: a Database Benchmark Based on the Facebook Social Graph. SIGMOD, 2013: 1185-1196.
[11] Cooper B F, Silberstein A, Tam E, et al. Benchmarking Cloud Serving Systems with YCSB. SOCC, 2010: 143-154.
[12] Liang F, Feng C, Lu X, et al. Performance Benefits of DataMPI: A Case Study with BigDataBench. BPOE, 2014: 111-123.
[13] Hoare C A R. Communicating Sequential Processes. Communications of the ACM, 1978, 21(8): 666-677.
[14] Si Y, Sun J, Liu Y, et al. Model Checking With Fairness Assumptions using PAT. Frontiers of Computer Science, 2014, 8(1): 1-16.

Formal Verification and Analysis of Time-Sensitive Software-Defined Network Architecture

Weiyu Xu[1], Xi Wu[2], Yongxin Zhao[1*], and Yongjian Li[3]
[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
[2] The University of Sydney, Australia
[3] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

***Abstract*—Safety-critical traffic in Industrial Internet of Things (IIoT) requires real-time communications with high fault tolerance, bounded latency and low jitter. Time-Sensitive Software-Defined Network (TSSDN), which combines the deterministic transmission of Time-Sensitive Networking (TSN) with the centralized management of Software-Defined Networking (SDN), was recently proposed to support the real-time requirement in IIoT. The research on TSSDN has been receiving increasing interests, however, the existing work has limitations including 1) the functional safety of TSSDN cannot be guaranteed; and 2) the effect of the separation of data plane and control plane on the time-sensitivity of TSSDN has not been evaluated. Therefore, in this paper, we employ the timed model checker UPPAAL to formalize the TSSDN architecture. Firstly, we use the build-in checker in UPPAAL to verify deadlock-free property, functional safety property and starvation-free property of our model. Then, the total latency of frames forwarding and scheduling within a single switch is measured based on the model. We focus on the latency overhead of frames requesting processing rules from the controller, which is on average an additioanl 180μs latency in the worst case, but the impact of this delay on the time-sensitivity of TSSDN is tolerable. As far as we know, this is the first paper providing a formal verification and analysis approach for TSSDN architecture, which could benefit for both TSSDN designers as well as the researchers.**

***Index Terms*—TSSDN Architecture, Real-Time Communication, Formalization, Verification, Timing Analysis**

I. Introduction

The fourth industrial revolution (i.e., industry 4.0) focuses on the integration of physical objects, humans and smart digital echnology in a sophisticated information network, which is also known as the Industrial Internet of Things (IIoT) [1]. With the rapid development of the IIoT, its safety-critical applications increasingly request both the real-time communication and run-time flexibility, which, however, cannot be both supported in the existing networking (e.g., Real-Time Ethernet) [2]. In 2016, Nayak et al. proposed the Time-Sensitive Software-Defined Network (TSSDN) [3], which integrates the deterministic and reliability of Time-Sensitive Networking (TSN) with the flexibility, heterogeneity and reconfigurability of Software-Defined Networking (SDN), to support the above requirements in the IIoT.

Recently, the research on TSSDN has been receiving increasing interests, which mainly focus on architecture design [4]–[6], control strategy [7]–[9] and latency analysis [10], [11]. Various solutions on how to combine TSN and SDN to form the TSSDN architecture have been discussed, among which our work is based on the one proposed by Böhm et al. [5]. Specifically, TSN and SDN utilize a unified control plane, therefore all network devices are managed by a centralized TSSDN controller. Based on the timing analysis evaluation, Nayak et al. [10] showed that TSSDN provided deterministic end-to-end latencies with low and bounded jitter for the time-triggered traffic on a specific benchmark topology. However, there are few work on the formalization of the TSSDN architecture, especially lacking the verification on its properties (e.g., safety and time-sensitivity) and formal analysis about how the separation of data plane and control plane affects the time-sensitivity of TSSDN.

In this paper, we use the model checker UPPAAL to formally verify and analyze the TSSDN architecture. We formalize the TSSDN controller and the switch as timed automata. Then, we verify the deadlock-free property, functional safety properties and the starvation-free property in our model via the UPPAAL build-in checker. Finally, we measure the total latency of frames forwarding and scheduling within a switch, and assess the additional latency of the frames requesting processing rules from the controller in the worst case. Our formal verification results illustrate that the TSSDN properties in the model are satisfied, and our timing analysis shows that the transmission over the TSSDN architecture is still time-sensitive to some extent. To the best of our knowledge, this is the first work on the formal verification and analysis of the TSSDN architecture.

The main contributions of this paper are:

1) **Formal Modeling.** We present a formal model of TSSDN architecture via timed automata in UPPAAL. Properties (e.g., functional safety property) can then be verified within the model. This approach can facilitate both researchers and designers to assess the time performance and validity of TSSDN architecture before deployment.
2) **Timing Analysis.** Our timing analysis mainly focuses on how the latency overhead caused by the separation of data panel and control panel affects the time-sensitivity in TSSDN. It is the first work on TSSDN architecture combining formal verification with analysis approaches.

The remainder of this paper is structed as follows. A brief introduction of the TSSDN architecture and model checker

*Corresponding author: yxzhao@sei.ecnu.edu.cn
DOI reference number: 10.18293/SEKE2022-094

UPPAAL has been given in Section II. In Section III, we present the formal model of the TSSDN architecture in UPPAAL. Section IV presents an experimental implementation with formal verification and timing analysis. Finally, Section V concludes this paper and presents the future work.

II. Preliminaries

A. TSSDN Architecture

As specified by the Open Network Foundation (ONF) [12], TSSDN architecture is divided into three layers, including the application plane, the control plane and the data plane, which can be found in Fig. 1. The application plane provides a

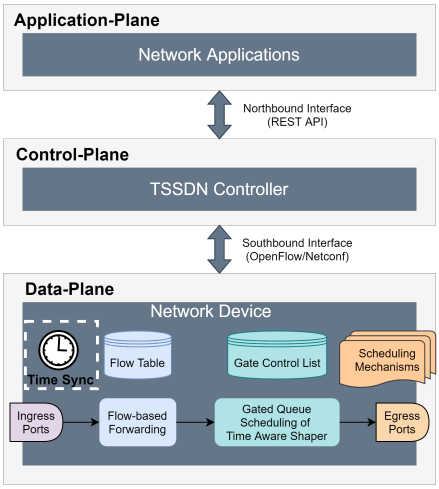


Fig. 1. TSSDN Architecture

programmable platform to users, which calls different services provided by the control plane through the northbound REST API. The control plane interacts with all network devices in the data plane to be aware of a global view of the network. According to the global state of the network, the centralized TSSDN controller can dynamically generate and distribute network configuration and control information to the data plane through the southbound API (e.g., Forces and OpenFlow [13]). The data plane consists of all network devices (i.e., bridges and switches). Its main responsibility is forwarding frames according to the rules in the flow table provided by the control plane.

Flow table is a finite set of flow table entries, which is used to control the forwarding of flows (i.e., sequences of frames with the same destination). According to the OpenFlow V1.0 [14], each flow table entry consists of $header\ fields$, $counters$ and $actions$. The head of each frame processed by the switch is compared against the $header\ fields$ of flow table entries. If a matching entry is found, any $actions$ for that entry will be performed on the frame (e.g., forward to a specified port, deliver to controller or drop). The controller is responsible for determining how to handle frames delivered from the switch. At the egress ports of network devices, the transmission order of different traffic will be determined based on different scheduling mechanisms to guarantee the QoS.

B. UPPAAL

UPPAAL [15], [16] is a well-known model checker, widely used in modeling, simulation and verification of Cyber Physical Systems (CPS), aerospace systems, real-time scheduling systems and other areas. In UPPAAL, a real-time system is modeled as a collection of timed automata with real-valued clocks. A timed automaton can be represented by a six tuple $M_{st} = (L, C, \Sigma, E, I, I_o)$, where

1) L is a set of locations;
2) C is the set of clocks;
3) Σ is a set of actions over edges, which could change the value of variables or clocks;
4) $E \subseteq L \times \beta(C) \times \Sigma \times 2^C \times L$ is a set of edges, where $\beta(C)$ is the set of conjunctions over simple conditions of clock constraints and 2^C is the set of clocks to be reset;
5) $I : L \rightarrow \beta(C)$ is a mapping of invariant on locations;
6) I_0 is the initial location.

TABLE I
CTL Operators

	Operator	Meaning	Operator	Meaning
Logical	&&	conjunction	||	disjunction
	!	negation	$\rightarrow$	implication
Temporal	[]	always	<>	eventually
	X	next	U	until
Path	A	all	E	exist

UPPAAL has a simulator for quantitative analysis and a verifier for model checking. The simulator validates the model via system execution, whereas the verifier checks properties such as safety and liveness described by Computational Tree Logic (CTL) via on-the-fly traversing the entire state space. As shown in TABLE I, logical, temporal and path operators in CTL can be used to formally describe system properties.

III. Modeling of the TSSDN Architecture

The overview model of TSSDN architecture, which can be found in Fig. 2, consists of generator model, flow table model, controller model and scheduler model. The formal models mentioned above are represented in UPPAAL as timed automata templates which are then instantiated as one or more processes as necessary, shown as follows:

```
system Generator_Process,
       Flow_Table_Process,
       Scheduler_Process,
       Controller_Process;
```

The generator process is used to simulate the frames that have entered the switch and wait for matching with the flow table. The flow table process and scheduler process, respectively, represent forwarding and scheduling within a single switch. The controller process communicates with the flow table process and handles frames that request processing rules.

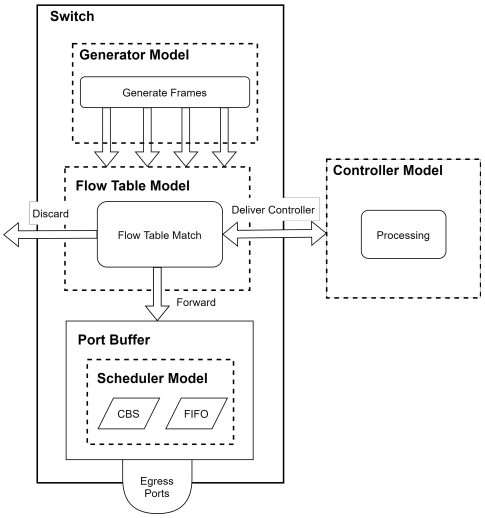


Fig. 2. An Overview Model of the TSSDN Architecture

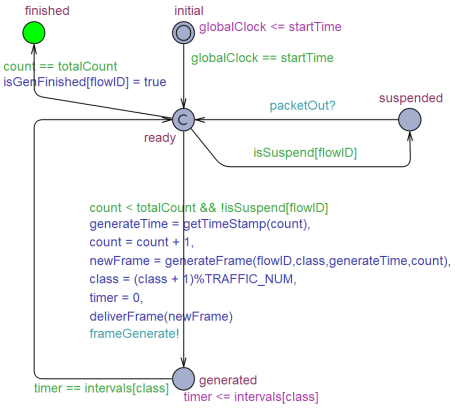


Fig. 3. Timed Automata of Generator Model

A. Generator Model

IEEE 802.1Qbv divides data frames into Control Data Traffic (CDT), Audio/Video Traffic (AVB_A and AVB_B) and Best Effort Traffic (BE) [17]. The frames of the above four classes will be generated and delivered to the flow table model on a regular basis in the generator model.

$Definition\ 1\ (Frame)$. A frame is represented by a quadruple ($flowID$, $class$, $transTime$, $timeStamp$), where

1) $flowID$ identifies the flow to which the frame belongs, that is, the abstraction of the destination (e.g., MAC address, IP address, or VLAN ID);
2) $class$ defines the class of the frame, i.e., CDT, AVB_A, AVB_B and BE;
3) $transTime$ represents the duration of the frame passing through ports. On the premise of a constant port transmission rate, it usually depends on the length of the frame;
4) $timeStamp$ stands for the generation time instant of the frame, which can be used for latency analysis.

We formalize the generator as a timed automata template in UPPAAL shown in Fig. 3. The automaton consists of five states: $initial$, $ready$, $generated$, $finished$ and $suspended$. The variable $timer$ is a timer used for generating frames on a regular basis and the variable $globalClock$ is a global clock used for the synchronization of all processes in the system. The variable $packetOut$ will be explained in the controller model. Other variables and functions are defined as follows:

1) $flowID$ should be specified when instantiating a generator process because $flowID$ of the frames generated by each generator is presumed to be the same;
2) $class$ stands for the traffic to which the frame to be generated belongs, and `TRAFFIC_NUM` specifies the number of traffic classes;
3) $totalCount$ represents the total number of frames to be generated;
4) $count$ is used to record the number of frames that have been generated;
5) $frameGenerate$ is a synchronization channel used to notify the flow table process to start matching the delivered frame with the flow table;
6) $intervals[...]$ denotes the generation intervals of different classes of frames;
7) $isSuspend[...]$ indicates whether the generator needs to be paused;
8) $isGenFinished[...]$ indicates whether all frames have been generated and delivered;
9) $generateFrame(...)$ is a function used to generate a new frame which will be stored in $newFrame$;
10) $deliverFrame(...)$ is a function, which is responsible for delivering a frame to the flow table model.

The generator process is initially in the $initial$ state. When $startime$ equals $globalclock$, it enters the $ready$ state and begins generating the first frame. After that, the process alternates between $ready$ state and $generated$ state to periodically generate and deliver frames. When the value of $isSuspend[...]$ is $true$, the process will switch to the $suspended$ state and suspend working. This is typically caused by frames delivered by this generator waiting for processing rules from the controller. The generator process will migrate to the $finished$ state when $count == totalcount$, indicating that all frames have been generated and delivered.

B. Flow Table Model

In the flow table model, frames arriving at the switch will be matched with entries in a pre-configured flow table, which is defined as follows:

$Definition\ 2\ (FlowTable)$. A flow table is composed of multiple flow table entries, each of which consists of a triple ($header\ field$, $counter$, $action$), where

1) $header\ field$ is used to match the $flowID$ of the frame;
2) $counter$ records the number of times a flow entry matches successfully;
3) $action \in \{\texttt{CONTROLLER}, \texttt{DROP}, \texttt{PORT\_A}, \texttt{PORT\_B}, ...\}$ indicates the action that the matched frame should take.

The timed automata template of the flow table model can be found in Fig. 4, which will be instantiated into a unique flow

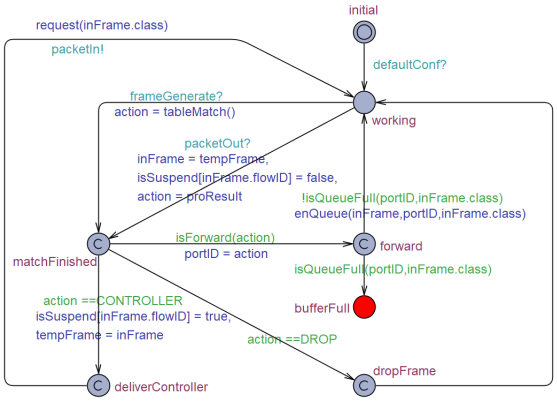


Fig. 4. Timed Automata of Flow Table Model

table process. After receiving the default flow table issued by the controller, the flow table process will move from $initial$ state to $working$ state. Whenever the process receives the synchronization signal $frameGenerated$ sent from generators, the delivered frame $inFrame$ will be matched with the flow table, then the process shifts to $matchedFinished$ state. According to the $action$ of the matched entry, the process will move to $forward$ state, $deliverController$ state or $dropFrame$ state. The forwarded frames will be placed in the matching egress port's buffer frame queues. Particularly, if the $action$ of the matched entry is `CONTROLLER`, $inFrame$ will be temporarily stored in $tempFrame$. Then the switch will notify the generator of the frame to suspend working and send a $packetIn$ message to the controller model requesting how to handle the frame. After receiving the $packetout$ message from the controller, $tempFrame$ will be discarded or forwarded to the specified egress port depending on the $proResult$ carried in $packetout$. Other variables and functions are defined as follows:

1) $tableMatch(...)$ is a function matching $inFrame$ with the flow table;
2) $isQueueFull(...)$ is a function checking the frame queue of a specific traffic in the port buffer is full. If the queue is full, the process will enter $bufferFull$ state;
3) $enQueue(...)$ is a function, responsible for enqueuing frames to the buffer queue of a specified port.

C. TSSDN Controller Model

The timed automata template of TSSDN controller can be found in Fig. 5. During the system initialization, instantiated controller process will send a default flow table to the switch and move to $working$ state. After receiving a $packetIn$ message from a switch, the controller will spend $maxProcessTime$ on calculating and generating a processing rule via function $processRule(...)$ for the requested frame based on the real-time global network state. The function can be defined by designers and the additional latency in the worst case can be measured based on $maxProcessTime$. Then the process will send a $packetOut$ message back to the switch to handle the frame waiting for the processing rule. It takes `PACKET_IN_TRANS_TIME` and `PACKET_OUT_TRANS_TIME` for $packetIn$ and $packetOut$ messages to pass through a port respectively. Here we assume that $inFrame$ is always included in these two messages. Therefore, the transmission time of these two messages depends on the maximum length of frames.

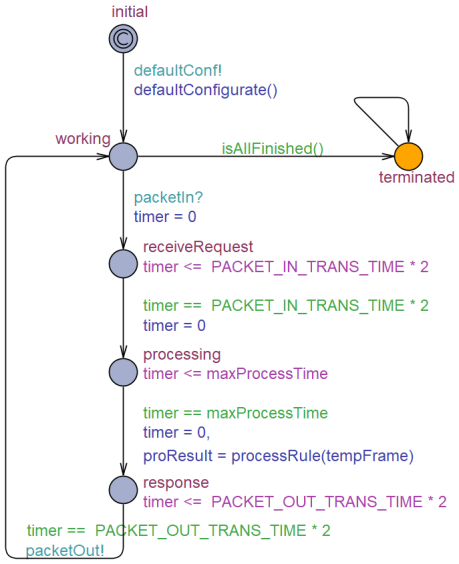


Fig. 5. Timed Automata of Controller Model

After scheduler process and all generator processes reach $finished$ state, controller process will move to $terminated$ state, indicating that the system has been ended. The remaining variables and functions are defined as follows:

1) $maxProcessTime$ represents the maximum processing time to calculate and generate processing rules, which is specified when instantiating the controller process;
2) $defaultConf$ is a synchronization channel used to active the flow table process;
3) $defaultConfigurate(...)$ is a function initializing system global variables and issuing the flow table.

D. Scheduler Model

According to IEEE 802.1Qbv, a scheduling period is divided into protected windows and unprotected windows. A guard band is introduced to prevent a new transmission of the non-time-critical traffic (i.e., AVB and BE traffic).

The timed automata template of the scheduler is shown in Fig. 6. A scheduler process will be employed to schedule frames according to different scheduling mechanisms for the frame queues at each egress port of the switch. The variables and functions are defined as follows:

1) $slots[...]$ is an array of time slots, whose values depend on the length of unprotected windows and protected windows;
2) $portID$ indicates the port associated with the scheduler;
3) $schClock$ is the clock of the scheduler, whose value is updated by function $updateClock(...)$;

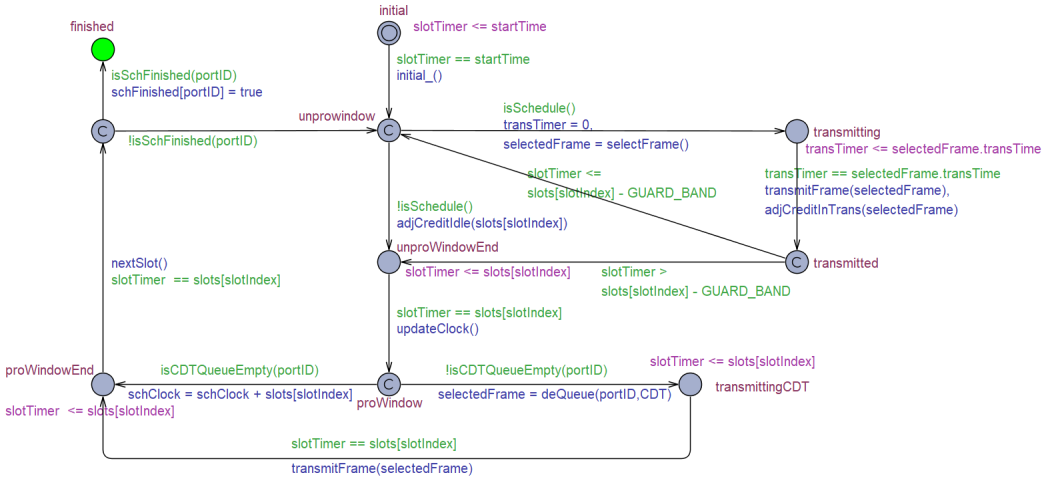


Fig. 6. Timed Automata of Scheduler Model

4) $schFinished[...]$ indicates whether all frames in the port buffer have been scheduled;
5) $selectFrame(...)$ is a key function used to schedule a frame from queues according to different scheduling mechanisms. CDT traffic is scheduled according to FIFO mechanism, whereas AVB traffic and BE traffic are scheduled based on credit-based shaping (CBS) mechanism, whose details are introduced in the Forwarding and Queuing enhancements for Time-Sensitive Streams (FQTSS) [18];
6) $adjCreditIdle$ is a function to update credits in CBS when no frame transmitted, while $adjCreditInTrans$ is to do so after a frame is transmitted;
7) $transitFrame(...)$ is a function used to calculate the latency of frames and to transmit frames through the egress ports of the switch;
8) $isSchedule(...)$ is a function to check whether scheduling a frame is allowed according to CBS mechanism.

When $slotTimer$ equals to $startTime$, the scheduler process transits from $initial$ state to $working$ state. Then the process enters three windows (i.e., unprotected, guard band, protected) in turn. CDT traffic can only be scheduled in protected windows, whereas AVB and BE traffic can be scheduled in unprotected windows. Once the process enters the guard band, it is not allowed to start the transmission of any frame. When all of the slots are exhausted, the process will return to the $working$ state and start a new cycle. Particularly, when all frames in the port buffer have been scheduled, the process will move to $finished$ state and stop scheduling.

IV. Formal Verification and Timing Analysis

In this section, we perform the formal verification and timing analysis over the formal model given in section III.

A. Experimental Configuration

We mainly focus on the transmission of four classes of data frames: CDT, AVB_A, AVB_B, and BE. The parameters of traffic classes are shown in Table II. For simplicity, the frame of each class is generated and delivered to the switch at a fixed interval and the transmission rate of the port is 100 Mbps in our experiment. In the table, $transTime$, which depends on the size of frame and the transmission rate of ports, indicates the transmission time of the frame through a switch port. $\beta^+$and β^- denote the increasing rate and the decreasing rate of credits in CBS mechanism, respectively.

TABLE II
Parameters of Traffic Classes

Class	CDT	AVB_A	AVB_B	BE
Length	300B	425B	275B	325B
transTime	18μs	34μs	22μs	26μs
Interval	100μs	30μs	40μs	40μs
Priority	4	3	2	1
β^+	-	3	4	-
β^-	-	4	3	-

In view of IEEE 802.1 TSN [19], [20], TSSDN period is set to 500μs in our experiment, which is divided into two unprotected windows and two protected windows. Note that the last 34 μs of each unprotected window is configured as a guard band, which is equal to the transmission time of the frame with maximum length.

In our experiment, four instantiated generator processes are employed to simulate four flows waiting to be processed by the switch. We instantiate a flow table process and two schedulers to simulate a switch with two egress ports (i.e., `PORT_1` and `PORT_2`). The flow table of the switch is shown in Table III. One controller process communicates with the flow table process and handles frames that request processing rules.

In particular, each frame of `FLOW_D` will be included in the $packetIn$ message to request processing rule from the controller. In order to minimize the average latency of `FLOW_D`, the controller process always decides to forward the frames of `FLOW_D` to the egress port with the fewest frames. Additionally, the length of $packetIn$ and $packetOut$ messages

TABLE III
FLOW TABLE OF THE SWITCH

Header Field	Counters	Action
FLOW_A	0	PORT_1
FLOW_B	0	PORT_2
FLOW_C	0	DROP
FLOW_D	0	CONTROLLER

is set as 450 bytes to accommodate the frame with maximum length, resulting in a transmission time of 36μs for both.

B. Formal Verification and Analysis

1) Formal Verification. We employ ten assertions to verify the model satisfies the following four properties.

Property 1: Deadlock-free Property

$A[\,]\ not\ deadlock$

This property asserts that the system will never progress into a deadlock situation, which is satisfied in our model.

Property 2: Starvation-free Property

$A <>\ scheduler.selectedFrame.class\ ==\ \texttt{CDT}$
$A <>\ scheduler.selectedFrame.class\ ==\ \texttt{AVB\_A}$
$A <>\ scheduler.selectedFrame.class\ ==\ \texttt{AVB\_B}$
$A <>\ scheduler.selectedFrame.class\ ==\ \texttt{BE}$

These assertions claim that frames of all classes will eventually be scheduled, whose results can be found in Fig. 7.

Property 3: Window Exclusive

$A[\,]\ scheduler.transmitting\ imply$
$\quad scheduler.selectedFrame.class! = \texttt{CDT}$
$A[\,]\ scheduler.transmittingCDT\ imply$
$\quad scheduler.selectedFrame.class == \texttt{CDT}$

The above assertions verify that CDT can only be scheduled in protected windows, which are satisfied in our model.

Property 4: Validity of Flow Table Matching

$A[\,]\ switch\_.dropFrame\ imply$
$\quad (action == \texttt{DROP}\ ||\ proResult == \texttt{DROP})$
$A[\,]\ switch\_.deliverController\ imply$
$\quad flowTable[matchedIndex][\texttt{ACTION}]$
$\quad\quad == \texttt{CONTROLLER})$
$A[\,]\ switch\_.forward\ imply\ (action == proResult$
$\quad ||\ action ==\ flowTable[matchedIndex][\texttt{ACTION}])$

These assertions claim that the switch always handles frames correctly according to the matched flow table entry. The variable $matchedIndex$ indicates the index of the matched flow table entry.

2) Timing Analysis. The latency of a frame within a switch, to be precise, is the time elapsed from the frame beginning to match the flow table entries to the last bit of the frame being transmitted from the egress port of the switch. We measure the latency of frames of three flows (i.e., FLOW_A, FLOW_B and FLOW_D) within the switch, whose results can be found in Fig. 8. Note that, frames of FLOW_C are not considered here because they will be discarded according to the flow table.

We can see from the figure that the latency of BE and AVB frames in the three flows is basically stable, and the latency of traffic with higher priority is lower overall. The latency of most BE and AVB frames in FLOW_D is higher than that in FLOW_A

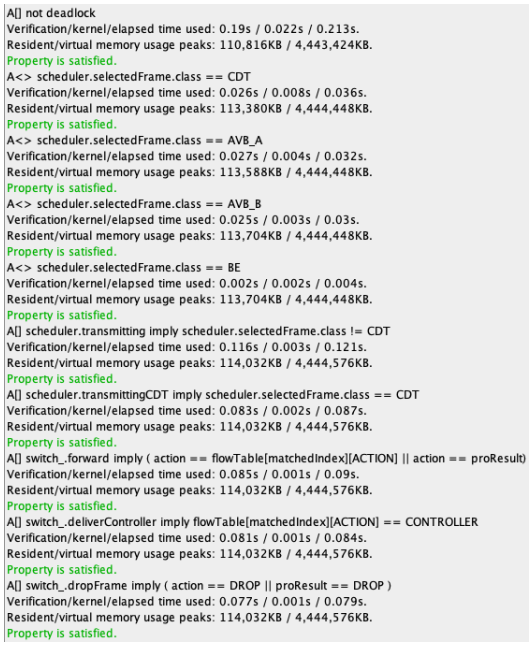


Fig. 7. Verification Results

TABLE IV
AVERAGE LATENCY OF FRAMES

Traffic\Flows	FLOW_A	FLOW_B	FLOW_C	FLOW_D
BE	205.2μs	219.4μs	-	330.8μs
AVB_B	128.6μs	137.5μs	-	329.4μs
AVB_A	128.8μs	103.5μs	-	340μs
CDT	626.3μs	820μs	-	447μs
Overall (exclude CDT)	154.2μs	153.6μs	-	333.4μs

and FLOW_B. However, the latency of CDT frames is much higher than that of frames of other traffic, and the latency of subsequent CDT frames continues to increase. Actually, this issue is caused by the division of time windows: there are only two protected windows in a scheduling cycle (i.e, a TSSDN period), and each protected window can only transmit one CDT frame. In other words, at most two CDT frames can be scheduled every 500μs, implying that a large number of CDT frames will be accumulated in the queue, resulting in a sharp increase in the latency of subsequent CDT frames.

Since our experiment mainly focuses on the additional latency of those frames that request the processing rules from the controller, CDT frames are excluded when calculating the overall average latency of the flows, which has little impact on our evaluation results. Table IV shows the average latency of various classes of frames in different flows, from which we find that the frames requesting the processing rule from the controller will incur an average of 180μs additional latency in the worst case.

V. CONCLUSION AND OUTLOOK

In this paper, we used the model checker UPPAAL to formally model and verify the TSSDN architecture. The results of verification demonstrate that the properties of the

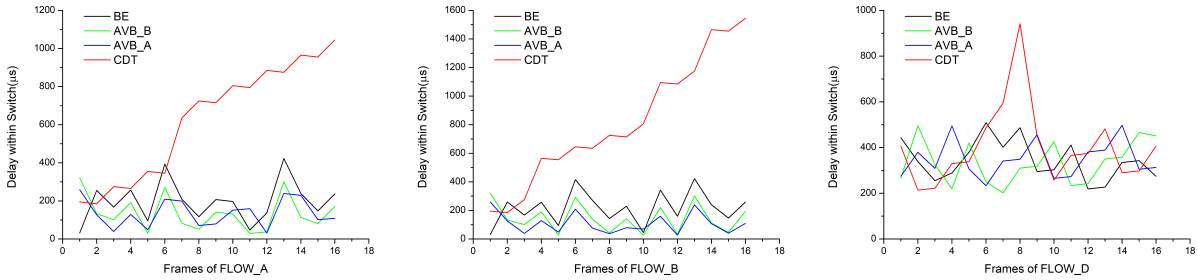


Fig. 8. Latency of Frames

TSSDN architecture are satisfied. Based on the model, we also performed a timing analysis and found that the frames requesting processing rules from the controller incurred an average of 180μs additional latency in the worst case, which shows that the transmission over the TSSDN architecture still retains time-sensitivity to some extent. By using our approach, both designers and researchers can conveniently verify the functional safety of the TSSDN architecture and assess the effect caused by the separation of data panel and control panel on time-sensitivity of TSSDN.

Limited by the size of the state space, our current model did not consider the process of frames entering the switch. State explosion is always a problem but can be mitigated by using abstraction and optimisation techniques (i.e. partial order reductions). [21] In the future, we would like to integrate the Per-Stream Filtering and Policing (PSFP) protocol into the model to capture the process by which frames are policed and filtered at ingress ports of the switch. Our future research will also focus on using various optimization techniques to enable scalability of the model.

ACKNOWLEDGEMENTS

This work is supported by Shanghai Science and Technology Commission Program under Grant 20511106002, Shanghai Trusted Industry Internet Software Collaborative Innovation Center and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[2] L. Silva, P. Pedreiras, P. Fonseca, and L. Almeida, "On the adequacy of sdn and tsn for industry 4.0," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2019, pp. 43–51.

[3] N. G. Nayak, F. Dürr, and K. Rothermel, "Software-defined environment for reconfigurable manufacturing systems," in *2015 5th International Conference on the Internet of Things (IOT)*. IEEE, 2015, pp. 122–129.

[4] M. Bhm, J. Ohms, O. Gebauer, and D. Wermser, "Architectural design of a tsn to sdn gateway in the context of industry 4.0," in *23. ITG-Fachtagung "Mobile Communications", ISBN: 978-3-8007-4577-7*, 2018.

[5] M. Böhm, J. Ohms, M. Kumar, O. Gebauer, and D. Wermser, "Time-sensitive software-defined networking: a unified control-plane for tsn and sdn," in *Mobile Communication-Technologies and Applications; 24. ITG-Symposium*. VDE, 2019, pp. 1–6.

[6] T. I. ul Huque, K. Yego, C. Sioutis, M. Nobakht, E. Sitnikova, and F. den Hartog, "A system architecture for time-sensitive heterogeneous wireless distributed software-defined networks," in *2019 Military Communications and Information Systems Conference (MilCIS)*. IEEE, 2019, pp. 1–6.

[7] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "An sdn architecture for time sensitive industrial iot," *Computer Networks*, vol. 186, p. 107739, 2021.

[8] T. Gerhard, T. Kobzan, I. Blöcher, and M. Hendel, "Software-defined flow reservation: Configuring ieee 802.1 q time-sensitive networks by the use of software-defined networking," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 216–223.

[9] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2017.

[10] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (tssdn) for real-time applications," in *International Conference*, 2016, pp. 193–202.

[11] D. Thiele and R. Ernst, "Formal analysis based evaluation of software defined networking for time-sensitive ethernet," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 31–36.

[12] O. N. Foundation. "sdn architecture". Tech. Rep. Issue 1, TR-502, 2014. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf/

[13] N. K. Haur and T. S. Chin, "Challenges and future direction of time-sensitive software-defined networking (tssdn) in automation industry," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 309–324.

[14] O. S. Consortium *et al.*, "Openflow switch specification version 1.0.0," *http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf*, 2009.

[15] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*. Springer, 2004, pp. 200–236.

[16] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal—a tool suite for automatic verification of real-time systems," in *International hybrid systems workshop*. Springer, 1995, pp. 232–243.

[17] J. Lv, Y. Zhao, X. Wu, Y. Li, and Q. Wang, "Formal analysis of tsn scheduler for real-time communications," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 1286–1294, 2020.

[18] I. S. Association *et al.*, "Ieee standard for local and metropolitan area networks—virtual bridged local area networks amendment 12 forwarding and queuing enhancements for time-sensitive streams," *IEEE Standard*, vol. 802, pp. 10 016–5997, 2009.

[19] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, pp. 1–1832, 2014.

[20] S. Thangamuthu, N. Concer, P. J. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 55–60.

[21] V. Klimis, G. Parisis, and B. Reus, "Towards model checking real-world software-defined networks," in *International Conference on Computer Aided Verification*. Springer, 2020, pp. 126–148.

Metaheuristic Algorithms for Proof Searching in HOL4

M. Saqib Nawaz[1], M. Zohaib Nawaz[2], Osman Hasan[3] and Philippe Fournier-Viger[1]
[1]School of Computer Science and Software Engineering, Shenzhen University, China
[2]Department of Computer Science and IT, University of Sargodha, Sargodha, Pakistan
[3]School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan
msaqibnawaz@szu.edu.cn, zohaib.nawaz@uos.edu.pk, osman.hasan@seecs.edu.pk, philfv@szu.edu.cn

***Abstract*—User guided proof development in interactive theorem proving is a manual and time consuming activity. For automating proof searching and optimization in a higher-order logic proof assistant, we provide two metaheuristic algorithms that are based on Fitness Dependent Optimizer (FDO) and Bat Algorithm (BA). In both metaheuristic algorithms, random proof sequences are first created from a population of frequently occurring proof steps that are discovered using pattern mining techniques. Created proof sequences are then evolved till their fitness matches the fitness of the original (or target) proof sequences. Experiments are performed to investigate the performance of the proposed algorithms on different HOL4 theories. Moreover, the proposed FDO and BA-based proof searching approaches are compared with Simulated Annealing (SA) and Genetic Algorithm (GA)-based methods. Results show that BA performs best, followed by FDO and SA for proof finding and optimization in HOL4.**

***Index Terms*—Proof Searching, HOL4, Fitness Dependent Optimizer, Bat Algorithm, Simulated Annealing, Genetic Algorithm**

I. Introduction

Interactive theorem provers (ITPs) are used not only for the formalization of mathematical theorems and substantial parts of theoretical computer science, but also to model and verify complex software and hardware systems [2]. In ITPs, the systems that need to be analyzed are first modeled using an appropriate mathematical logic. Important (and sometimes critical) system properties are then proved using *theorem provers* [6]. ITPs are generally based on higher-order logic (HOL). The rich logical formalisms offered by HOL enable ITPs to define and reason about complex systems. However, due to the undecidability in HOL, the reasoning process cannot be made fully automated and human guidance is always required in the process of proof searching and development [11]. Due to this, ITPs are also called *proof assistants*. Some widely used proof assistants are HOL4 [19], Isabelle/HOL [16], Coq [3] and PVS [17].

The user driven proof development process makes the proof guidance and automation as well as automatic proof searching some extremely desirable features for ITPs. Evolutionary and heuristic algorithms, also indicated in [7], [9], [20], can be used to efficiently search for the proofs of theorems/lemmas because of their ability and suitableness to handle black-box search and optimization problems. Thus for the HOL4 proof assistant, we proposed an evolutionary approach [15], where a Genetic Algorithm (GA) was used for proof searching and optimization. Moreover, a simulated annealing (SA)-based proof searching approach was developed [13], which outperformed the GA-based method.

Both proof searching approaches [13], [15] were found to be quite efficient in evolving random proofs. However, alternative proof searching approaches could be developed as a plethora of evolutionary/heuristic techniques are present in the literature. Thus, we further investigate the applicability of evolutionary/heuristic techniques in the HOL4 proof assistant. This paper extends prior works [13], [15] by proposing two more metaheuristic-based approaches, where Fitness Dependent Optimizer (FDO) [1] and Bat Algorithm (BA) [21] are used for proof searching and optimization in HOL4. The performance of FDO and BA are compared with that of GA [15] and SA [13] for various parameter values. Through experiments on proof sequences of formalized theorems/lemmas in different HOL4 theories, it is found that BA performs better than the other three algorithms, followed by FDO and SA. Whereas, different versions of GA perform poorly for proof finding and optimization.

II. Related Work

Some work has been done in the past where evolutionary algorithms were used in ITPs. For example, a GA was used to automatically find the formal proofs of theorems/lemmas in the Coq proof assistant [7], [20]. But a major limitation of this approach is that even though it can find small proofs for theorems that contain a few proof steps, a user is still required to interact with Coq to guide the proof process for large theorems/lemmas that contain more proof steps. The work [9] briefly discussed how evolutionary computation can be used to improve the heuristics of automatic proof search in Isabelle/HOL. The objective is to find heuristics that can select the most promising PSL [10] (proof strategy language

DOI reference number: 10.18293/SEKE2022-103

for Isabelle/HOL) strategy from various available hand written strategies when applied to a given proof goal.

Another work [4] used genetic programming [8] and a pairwise combination (that focused only on crossover-based approach) to evolve frequent proofs patterns in the Isabelle proof assistant into compound tactics. However, a linearized tree structure was used to represent Isabelle's proofs. The linearization sometimes leads to the loss of important connections (information) among different branches in the proof trees. Due to this, the evolution process may not find interesting patterns and tactics in those trees.

The proposed proof searching approaches, presented in this paper, overcome the aforementioned limitations as they can handle proof goals of various lengths. Moreover, the dataset of proof sequences has all the important information that is needed for the determination of frequent proof steps, through which an initial population is generated. Lastly, the proposed approaches do not require any sort of human guidance in the evolution process for random proof sequences.

III. Proposed Proof Searching Approaches

HOL4 follows the interactive proof development process using the *lambda calculus proof representation*. Formal proofs in HOL4 can be constructed with an interactive *g*oal stack that are then put together using the ML function *prove*. A user of HOL4 interacts with the proof assistant to guide the proof process by providing necessary tactics, definitions, and already verified theorems. HOL4 also offers automatic proof procedures that help the user in directing the proof.

To use evolutionary/heuristic algorithms for proof searching and optimization, the data available in HOL4 proof files is first converted to a proper computational format. Moreover, the redundant information (related to HOL4) that plays no part in proof searching and evolution is removed from the proof files. Now, the complete proof for a theorem/lemma is a sequence of *HPS* (HOL4 proof step).

Let PS = $\{HPS_1, HPS_2, \ldots, HPS_m\}$ denote the set of *HPS*. PSS, a *proof step set*, is a set of *HPS*, i.e., $PSS \subseteq PS$. For example, consider that PS = {*DISCH_TAC, REPEAT_GEN_TAC, RW, FULL_SIMP_TAC, PROVE_TAC, REWRITE_TAC*}. The set {*FULL_SIMP_TAC*,*RW*,*DISCH_TAC*,*REWRITE_TAC*} is a proof step set that contains four *HPS*. A proof sequence is a list of PSS's, i.e., $S = \langle PSS_1, PSS_2, ..., PSS_n \rangle$, such that $PSS_i \subseteq PSS$ $(1 \leq i \leq n)$. For example, $\langle$ {*FULL_SIMP_TAC, PROVE_TAC*}, {*DISCH_TAC, REPEAT_GEN_TAC, REWRITE_TAC*}, {*RW*}$\rangle$ is a proof sequence containing three PSS and six *HPS*.

A *proof dataset* PD is a list of proof sequences, i.e., $PD = \langle S_1, S_2, ..., S_p \rangle$. Each sequence in the PD has an identifier (ID) denoted as p. For example, Table I shows a *PD* that has five proof sequences.

Table I
A sample proof dataset

ID	Proof Sequence
1	$\langle$ {*GEN_TAC, Q_TAC, SUFF_TAC*}$\rangle$
2	$\langle$ {*Q_TAC, SRW_TAC, HO_MATCH_MP_TAC*}$\rangle$
3	$\langle$ {*RW, AP_TERM_TAC, MAP_EVERYTHING_TAC, CONJ_TAC, PROVE_TAC*}$\rangle$
4	$\langle$ {*CASES_TAC, DISCH_TAC, SUBGOAL_THEN, CASES_ON, BETA_TAC, AP_TERM_TAC, GEN_TAC*}$\rangle$
5	$\langle$ {*SRW_TAC, Q.SUBGOAL_THEN, SUBST1_TAC, RW_TAC, Q.EXISTS_TAC, FULL_SIMP_TAC*}$\rangle$

A. Proposed FDO

The Fitness dependant optimizer (FDO) [1] is motivated by the swarming behavior of bees during reproduction when they explore and look for new hives. FDO consists of two processes: (1) The scout bee searching process, and (2) The scout bee movement process. In the first process, the scout bees search for a suitable solution. In the second process, a random walk and a fitness weight is used to move a scout bee towards a new position that indicates a potentially better solution.

Algorithm 1 shows the pseudocode of the proposed FDO for proof finding and optimization in the HOL4 theories. FDO first creates an initial population (*Pop*) from frequent *HPS* (*FHPS*) that are discovered with sequential pattern mining (SPM) techniques [5]. From the initial population, a random scout bee (SB) is generated. The fitness of the solution, a target proof sequence (P), and SB is calculated with the fitness procedure listed in Algorithm 2. The fitness values guide the FDO towards the best solution(s) (proof sequences). Fitness evaluates the closeness (or similarity) of a given solution (SB) with the best solution (the target solution). The fitness value in this work denotes the total number of those positions in *SB* and P where *HPS* are same.

In FDO, the general equation to calculate the movement of a scout bee is:

$$x_{i,t+1} = x_{i,t} + pace \tag{1}$$

where $x_{i,t}$ represents the current scout bee at iteration (t) and the movement rate is denoted by $pace$ that sets the scout bee direction. The fitness value (fw) manages the $pace$:

$$fw = \left| \frac{x^*_{i,t,f}}{x_{i,t,f}} \right| \times wf \tag{2}$$

where $x*_{i,t,f}$ and $x_{i,t,f}$ denote the best global fitness of the solution, and the current fitness of the scout bee and wf is a weight factor that can be either 0 or 1. In our case, the best global fitness is the fitness of the target proof sequence (P) and the current fitness is the fitness of the current scout bee (SB). Moreover, wf is set to 1 as FDO was unable to evolve random scout bees to target proof sequences when wf = 0.

FDO considers some scenarios for fw to provide a random mechanism for the $pace$. For example, if fw = 1 or 0, and $x_{i,t,f}$ = 0, FDO uses Equation (3) to find the $pace$ randomly. On the other hand, if $0 < fw < 1$, then FDO generates a random number r, in the range [-1, 1], to make sure that the scout bee searches in every direction. For different values of r, the $pace$ is calculated using equation (4):

$$pace = (x_{i,t} \times r) \qquad if((fw = 1 \wedge 0) \wedge x_{i,t,f} = 0) \tag{3}$$

Algorithm 1 FDO proof finding

Input: *FHPS*: Frequent HOL4 proof steps, *PD*: proof sequences database
Output: Generated proof sequences

Pop ← *FHPS*
for each $P \in PD$ **do**
  $g_b \leftarrow Fitness(P, P)$
  $SB \leftarrow$ randomseq(*Pop*, length(*P*))
  $p_b \leftarrow Fitness(SB, P)$
  $FS \leftarrow ()$
  if $p_b = g_b$ **then**
    return SB
  end if
  while $(p_b < g_b)$ **do**
    for i in range(length(SB)) **do**
      if $SB[i] = P[i]$ **then**
        $FS.append[i]$
      end if
    end for
    Calculate movement with $pace$ using Equation (5)
    $NS \leftarrow update\_position(SB, pace, FS)$
    $NF \leftarrow Fitness(NS, P)$
    if $NF = g_b$ **then**
      return NS
    end if
    if $NF > p_b$ **then**
      $SB \leftarrow NS$
      $p_b \leftarrow NF$
    end if
  end while
  return SB
end for

$$pace = \begin{cases} (x_{i,t} - x*_{i,t})fw(-1) & \text{if } 0 < fw < 1 \vee r < 0 \\ (x_{i,t} - x*_{i,t})fw & \text{if } 0 < fw < 1 \vee r \geq 0 \end{cases} \quad (4)$$

Algorithm 2 Fitness

Input: *Pseq*: A proof sequence, *P*: The current target proof sequence
Output: Integer that represents the fitness of a proof sequence (*Pseq*)

procedure FITNESS(*Pseq, P*)
  $i, f \leftarrow 0$
  while ($i \leq$ length(*Pseq*) - 1) **do**
    if (*Pseq*[i] = *P*[i]) **then**
      $f \leftarrow f + 1$
    end if
    $i \leftarrow i + 1$
  end while
  return f
end procedure

According to the nature of our problem, the movement in Equation (1) is adapted to be an integer number. Thus, Equation (1) is modified as:

$$x_{i,t+1} = x_{i,t} + \lfloor pace \rfloor \quad (5)$$

where $\lfloor pace \rfloor$ returns the integer that is less than or equal to $pace$.

Equation (5) for updating the movement basically indicates how many positions are required to be changed in a random proof sequence (SB) so that it reaches the next position. In the position update process, randomly selected position values in a scout bee are changed from their original values.

Algorithm 3 Update_Position

Input: PS: A proof sequence, UP: updated position, and FS: $fixedSlots$ array
Output: Updated Sequence

procedure UPDATE_POSITION(PS, UP, FS)
  for i in range$(0, UP)$ **do**
    $rp \leftarrow$ randomint(1, length(PS))
    if ($rp \notin FS$) **then**
      $alter \leftarrow$ randomsample(*Pop*, 1) ▷ (1 HPS form Pop)
      $PS[rp] \leftarrow alter$ ▷ ($PS[rp] \neq$ alter)
    end if
  end for
  return PS
end procedure

We use an array called *fixedSlots*(FS) to further enhance the searching process in FDO. This array keeps track of each position in SB that has matched its value with the P. During the update of SB positions, it is checked whether each position in SB that is to be replaced with a random *HPS*, is already present in *FS* or not. If the position is present in FS, then another random number is generated for a different position. If the position is not present, then that particular position is updated with a random HPS from *Pop*. Algorithm 3 is the procedure for updating the position.

B. Proposed BA

Bat Algorithm (BA) [21] is inspired by the echolocation behavior of microbats, with varying pulse rates of emission and loudness. The three main steps of BA are: (1) Estimating the optimal distance of bats towards the solution(s) using the phenomena of echolocation, (2) Bats moving in the search space with distinct velocity and fixed frequency. The wavelength and loudness can vary according to bats distance between solution(s) and the bat current position, and (3) Linearly decreasing the loudness and increasing the emission factor of bats when they are near to the solution(s). Algorithm 4 shows the pseudocode of the proposed BA for proof searching and optimization in HOL4 theories.

BA also creates an initial population (*Pop*) first from *FHPS*. A random proof sequence, that represents a bat (B), is then generated from the population. In general, the BA uses the following equations to calculate the frequency, velocity, and position of a bat, respectively:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (6)$$

$$v_i^{t+1} = v_i^t + (x_i^t - X_*)f_i \quad (7)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (8)$$

where f_i is the frequency of the i-th bat, f_{max} and f_{min} represent the maximum and minimum frequencies of the sound waves released by bats, and β is a random number in the range [0, 1]. Moreover, v_i^t and v_i^{t+1} represent the velocities of the i-th bat at iterations t and t + 1, respectively, x_i^t and x_i^{t+1}

Algorithm 4 BA proof finding

Input: *FHPS*: Frequent HOL4 proof steps, *PD*: proof sequences database
Output: Generated proof sequences

Pop ← *FHPS*
for each $P \in PD$ **do**
  $g_{best} \leftarrow Fitness(P, P)$
  $B \leftarrow$ randomseq(*Pop*, length(*P*))
  $p_{best} \leftarrow Fitness(B, P)$
  $FS \leftarrow ()$
  if $p_{best} = g_{best}$ **then**
    return B
  end if
  while $(p_{best} < g_{best})$ **do**
    for i in range(length(B)) **do**
      if $B[i] = P[i]$ **then**
        $FS.append[i]$
      end if
    end for
    Calculate updated velocity (v_i^{t+1}) using Equation (11)
    $NS1 \leftarrow update\_position(B, v_i^{t+1}, FS)$
    Calculate y_i^{t+1} using Equation (12)
    $ran1 \leftarrow rand(0, 1)$
    if $ran1 < y_i^{t+1}$ **then**
      $NS \leftarrow$ Neighbor($NS1$)
    else
      NS ← NS1
    end if
    $NF \leftarrow Fitness(NS, P)$
    if $NF = g_{best}$ **then**
      return NS
    end if
    if $NF > p_{best}$ **then**
      $B \leftarrow NS$
      $p_{best} \leftarrow NF$
    end if
  end while
  return B
end for

represent the locations of the i-th bat at iterations t and t + 1, respectively, and X_* represents the current optimal best location. In this work, X_* is equal to the total number of *HPS* in the target proof sequence.

While approaching towards the prey (target proof sequence in this work), a bat increases its pulse emission rate and decreases its loudness. These phenomena are simulated using the following equations:

$$A_i^{t+1} = \alpha A_i^t \tag{9}$$

$$r_i^{t+1} = r_i^0(1 + exp(\gamma t)) \tag{10}$$

where A_i^t and A_i^{t+1} represent the loudness at iterations t and t + 1, respectively, r_i^0 represents the initial pulse emission rate, r_i^{t+1} represents the pulse emission rate at iteration t + 1 and $0 < \alpha < 1$ and $\gamma > 0$ are constants.

For proof searching and optimization, a random bat (B) updates its velocity, location, loudness, and pulse emission rates repeatedly, until the target proof sequence (P) is reached.

Similar to movement update for FDO, the velocity update Equation (7) for BA is rewritten as:

$$v_i^{t+1} = v_i^t + \lfloor (gBest - x_i^t) f_i \rfloor \tag{11}$$

where $gBest$ is equal to X_* in Equation (11) and $\lfloor (gBest - x_i^t) f_i \rfloor$ returns the integer that is less than or equal to $(gBest - x_i^t)$.

Equation (11) for updating the velocity in BA indicates how many positions are required to be changed in a random proof sequence (B) so that it reaches the next position. Algorithm 3 is also used in BA to update the velocity position.

Equations (9) and (10) are used to calculate the loudness and pulse emission of a bat (B). Here, we combine these two equations together as follows:

$$y_i^{t+1} = A_i^{t+1} + r_i^{t+1} \tag{12}$$

Using the BA idea, if a random number $ran1$ is less than y_i^{t+1}, then one position value is changed in bat B. Algorithm 5 lists the procedure for changing one position in B. The randomly selected location value (*HPS*) is changed from its original value using the $Neighbor$ procedure in Algorithm 5.

Algorithm 5 Neighbor

Input: P_1: A proof sequence
Output: The neighbor proof sequence

procedure NEIGHBOR(P_1)
  $ind \leftarrow$ randomint(1, length(P_1))
  $alter \leftarrow$ randomsample(*Pop*, 1) ▷ (*1-HPS* form *Pop*)
  $P_1[ind] \leftarrow alter$ ▷ ($P_1[ind] \neq$ alter)
  return P_1
end procedure

IV. Results and Discussion

The proposed FDO and BA are implemented in Python[1]. To evaluate the proposed approaches, experiments were carried on a computer equipped with a fifth generation Core $i5$ processor and 4 GB of RAM. For FDO, the weight factor wf is set to 1. For BA, f_{min} and f_{max} are set to 0 and 10, respectively. Whereas, r_i^0, the initial loudness (A_i^0), α and γ are set to 0.2, 1, 0.8 and 0.9, respectively.

Table II
A sample of theorems / lemmas in some HOL4 theories

HOL Theory	No.	HOL4 Theorems / Lemmas
Transcendental	L1	⊢ ∀x. 0<=x∧x <= inv(2) ==> exp(x) <= 1+2*x
	T1	⊢ ∀ x. (\n. (^exp_ser) n*(x pow n)) sums exp(x)
	T2	⊢ ∀ x. 0 < x∧ x < 2 ==> 0 < sin (x)
Arithmetic	T3	⊢ ∀n a b. 0 < n ==>((SUC a MOD n = SUC b MOD n) = (a MOD n = b MOD n))
RichList	T4	⊢ ∀m n. ((l:'a list). ((m + n)=(LENGTH l))==> (APPEND (FIRSTN n l) (LASTN m l) = l)
Number	T5	⊢ ∀n m. (m <= n ==> (iSUB T n m = n - m)) ∧ (m < n ==> (iSUB F n m = n - SUC m))
	T6	⊢ ∀ n a. 0 < onecount n a ∧ 0 < n ==> (n = 2 EXP (onecount n a - a) - 1)
Sort	T7	⊢(PERM L[x]<==>(L= [x])∧(PERM [x] L <==>(L = [x])
	T8	⊢ PERM = PERM_SINGLE_SWAP
Rational	T9	⊢ ∀ x y. abs_rat (frac_add (rep_rat (abs_rat x)) y) = abs_rat (frac_add x y)
	T10	⊢ ∀ r1 r3. rat_les r1 r3 ==> ?r2. rat_res r1 r2 ∧ rat_les r2 r3

[1]Code available at: https://github.com/saqibdola/BAFDO-HOL4

We examined the performance of proposed FDO and BA in finding the correct proofs of theorems/lemmas in 14 different HOL4 theories available in its library. These theories are: *Transcendental, Arithmetic, RichList, Number, Sort, Rational, Bool, FiniteMap, InductionType, BinaryWords, Encode, Coder, Decode* and *Combinator*. From each theory, five to twenty theorems/lemmas were randomly selected. The *PD* contains 300 proof sequences in total and 93 distinct *HPS*. Some important theorems/lemmas from aforementioned theories are listed in Table II. Table III shows the performance of the FDO and BA on theorems/lemmas that are listed in Table II.

Recently, we used a GA [15] with various crossover and mutation operators and a SA [13] for proof searching and optimization in HOL4. Just like FDO and BA, an initial population for GA and SA was first created using the SPM-based learning approach [14]. Crossover and mutation operators were used in GA and annealing process in SA to evolve the random proof sequences towards the original (target) proofs. In GA, three crossover operators (single point crossover (SPC), multi point crossover (MPC) and uniform Crossover (UCO)) and two mutation operators (standard mutation (SM) and modified pairwise interchange mutation (MPIM)) were used. The main reason to use different versions of crossover and mutation operators was to compare their effect on the overall performance of GAs in proof searching.

We run the algorithms for GA and SA on *PD* to compare its performance with FDO and BA. The comparison of FDO, BA with SA and GA for the theorem ($T2$) is shown in Table III. For $T2$, BA performed better (8,017 generations) than others, followed by FDO (16,767 generations) and SA (30,346). For GA, we found that using different crossover operators has no major effect on its overall performance. However, MPIM was faster to find the target proof sequences than SM.

Table III
RESULTS FOR FDO, BA AND COMPARISON WITH GA, SA

T/L	Algorithm	Fitness	Generations	Time (s)
L1		54	10,027	0.362
T1		58	10,809	0.385
T2		**81**	**16,767**	**0.793**
T3		66	11,887	0.475
T4		19	4,073	0.062
T5		23	4,858	0.071
T6	FDO	20	3,880	0.069
T7		17	3,186	0.0417
T8		42	6,959	0.166
T9		23	4,594	0.062
T10		23	4,436	0.060
L1		54	5,885	0.251
T1		58	6,124	0.273
T2		**81**	**8,017**	**0.506**
T3		66	6,414	0.272
T4		19	1,974	0.052
T5		23	2,335	0.075
T6	BA	20	2,921	0.080
T7		17	1,758	0.019
T8		42	4,487	0.098
T9		23	3,012	0.078
T10		23	2,894	0.082
T2	SA	81	30,346	0.858
T2	GA(SPC/SM)	81	2,231,664	58.56
T2	GA(MPC/SM)	81	2,713,867	69.84
T2	GA(UC/SM)	81	2,905,410	75.63
T2	GA(SPC/MPIM)	81	500,500	14.89
T2	GA(MPC/MPIM	81	524,272	16.14
T2	GA(UC/MPIM)	81	589,292	17.15

The average number of generations for the four algorithms to reach the target proof sequences in the whole dataset are shown in Table IV. BA performed better than other algorithms, followed by FDO, SA and GA. The possible reason for this is that the $fixedSlots$ array in FDO and BA ensures that no changes are made in those positions where the HPS in both random solutions (bee in FDO and bat in BA) and the target solution (original proof sequence) match. This prevents the mismatching of HPS at already matched positions in both solutions. The reason for BA performing better than FDO is that besides the update velocity function, the random bat in BA also goes through the $Neighbor$ procedure that allows more diversity. GA with different crossover and MPIM operators is approximately fourteen times faster (generation wise) than GA with different crossover operators and SM. This is due to the fact that the SM changes the *HPS* at a single location in the sequence, whereas MPIM changes two locations. Thus, MPIM explores a more diverse solution as compared to SM. On the other hand, SA is six times faster than GA with MPIM and different crossover operators. Whereas, FDO is approximately one and seven times faster than SA, and BA is approximately twice faster than FDO. Moreover, the memory used by proof searching approaches while searching for proofs of formalized theorems/lemma in *PD* is also listed in Table IV. All the algorithms require approximately the same memory with negligible difference.

Table IV
AVERAGE TOTAL GENERATION COUNT FOR FDO, BA, SA AND GA

	Avg. Generation Count	Total Time	Memory
FDO	779,819	14.15 s	3521 Mb
BA	375,044	9.09 s	3454 Mb
SA	1,319,745	19.54 s	3459 Mb
GA(SPC/SM)	123,513,780	1844.50 s	3545 Mb
GA(MPC/SM)	120,580,649	1697.47 s	3463 Mb
GA(UC/SM)	119,633,993	1569.69 s	3507 Mb
GA(SPC/MPIM)	8,833,888	194.25 s	3550 Mb
GA(MPC/MPIM)	9,141,943	208.34 s	3702 Mb
GA(UC/MPIM)	8,704,233	190.491 s	3682 Mb

The longest proof in the *PD* is for theorem $T2$ (positive value of sine) and it consists of 81 *HPS*. Here we call this theorem *PVoS*. We checked how much time the four algorithms take generation wise and also how many correct *HPS* in *PVoS* are found in different generations by the algorithms. The lines in Figure 1 represent the time for algorithms and the bars represent the fitness achieved by the algorithms. BA reached the maximum fitness of 81 within approximately 8,000 generations. As generation increases, the performance of algorithms tend to decrease for fitness. This means that with more generations, algorithms were slow in finding the correct *HPS* for a proof sequence as compared to earlier generations. An interesting behavior for GA is that it tends to decrease the fitness values (found *HPS*) in some generations. For example, GA(MPC/MPIM) and GA(MPC/SM) found less *HPS* at 11,000 and 13,000, respectively, compared to correctly found *HPS* at earlier locations (10,000 and 12,000 respectively). The other algorithms do not exhibit such behavior.

Lastly, the four algorithms are compared in terms of convergence speed to examine how fast the algorithms were able to

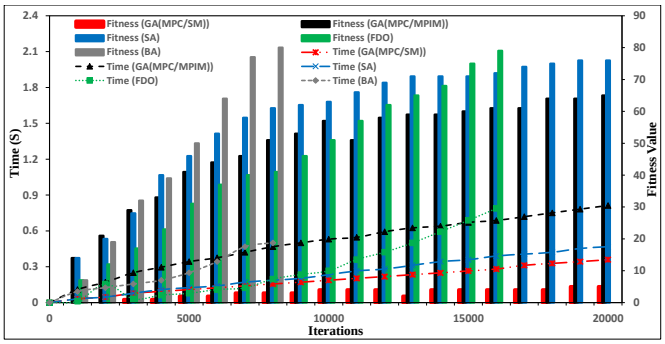


Figure 1. Time and fitness in different generations

converge towards the optimal solution. For the first 20,000 generations, the convergence speed of the four algorithms for *PVoS* is shown in Figure 2. BA converges very fast and found the correct *HPS* in approximately 8,000 generations. The performance of GA(MPC/MPIM) and SA was same compared to BA at the start. However, after 5,000 generations, GA(MPC/MPIM) and SA took more generations in finding the remaining correct HPS. The performance of SA after 13,000 generations tends to get low. Whereas, FDO performance was linear and fast from the beginning. On the other hand, GA(MPC/SM) convergence speed is slow from the start. At 20,000 generations, GA(MPC/MPIM) finds approximately 64 correct *HPS*, GA(MPC/SM) finds approximately 5 correct *HPS*, whereas SA finds 76 correct *HPS*.

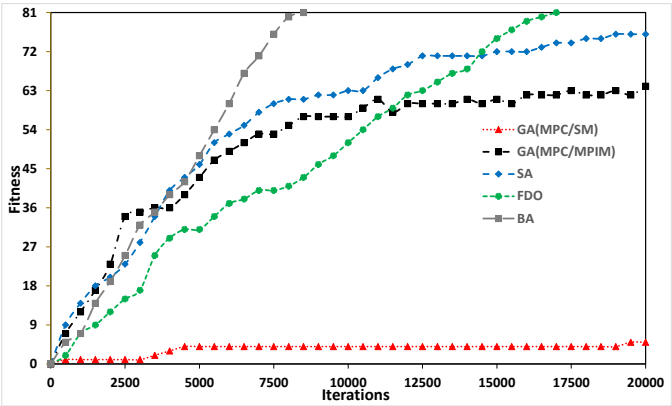


Figure 2. Convergence performance

In summary, it was observed through experiments that the proposed FDO-based and BA-based proof searching approaches can quickly optimize and automatically find the correct proofs for formalized theorems/lemmas in HOL4 theories. The proof searching approaches in this work and in [12], [13], [15] are not limited to HOL4 and can be used in proof assistants such as Isabelle/HOL [16], Coq [3], and PVS [17].

V. Conclusion

Despite huge developments in the last two decades, ITPs still depend on user interaction to manually guide proof assistants in finding the proof for a conjecture (unproved theorem or lemma). This interaction makes the proof development process quite complicated and a time consuming activity for the users. This paper introduced two proof searching approaches based on FDO and BA for optimizing and finding the correct proofs in various HOL4 theories. Additionally, a performance comparison of the two approaches with SA and GA showed that both FDO and BA performed better than them.

In future, we are interested in exploiting the Curry-Howard correspondence in sequent calculus [18] that offers a relationship between programming and mathematical proofs. This will allow us to use evolutionary/heuristic techniques to write programs (proofs) and use HOL4 proof assistant to simplify and verify by computationally evaluating the programs.

References

[1] J. M. Abdullah and T. Ahmed. Fitness dependent optimizer: Inspired by the bee swarming reproductive process. *IEEE Access*, 7:43473–43486, 2019.

[2] J. Avigad, J. C. Blanchette, G. Klein, L. C. Paulson, A. Popescu, and G. Snelting. Introduction to milestones in interactive theorem proving. *Journal of Automated Reasoning*, 61(1-4):1–8, 2018.

[3] Y. Bertot and P. Casteran. *Interactive theorem proving and program development: Coq'Art: The calculus of inductive construction*. Springer, 2003.

[4] H. Duncan. *The use of data-mining for the automatic formation of tactics*. PhD thesis, University of Edinburgh, UK, 2007.

[5] P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[6] O. Hasan and S. Tahar. Formal verification methods. In *Encyclopedia of Information Science & Technology, 3rd edition*, pages 7162–7170. IGI Global, 2015.

[7] S. Y. Huang and Y. P. Chen. Proving theorems by using evolutionary search with human involvement. In *Proceedings of CEC 2017*, pages 1495–1502. IEEE, 2017.

[8] J. R. Koza. *Genetic programming - On the programming of computers by means of natural selection*. MIT Press, 1993.

[9] Y. Nagashima. Towards evolutionary theorem proving for Isabelle/HOL. In *Proceedings of GECCO (Poster) 2019*, pages 419–420. ACM, 2019.

[10] Y. Nagashima and R. Kumar. A proof strategy language and proof script generation for Isabelle/HOL. In *Proceedings of CADE 2019*, volume 10395 of *LNCS*, pages 528–545. Springer, 2017.

[11] M. S. Nawaz, M. Malik, Y. Li, M. Sun, and M. I. U. Lali. A survey on theorem provers in formal methods. *CoRR*, abs/1912.03028, 2019.

[12] M. S. Nawaz, M. Z. Nawaz, O. Hasan, P. Fournier-Viger, and M. Sun. An evolutionary/heuristic-based proof searching framework for interactive theorem prover. *Applied Soft Computing*, 104:107200, 2021.

[13] M. S. Nawaz, M. Z. Nawaz, O. Hasan, P. Fournier-Viger, and M. Sun. Proof searching and prediction in HOL4 with evolutionary/heuristic and deep learning techniques. *Applied Intelligence*, 51(3):1580–1601, 2021.

[14] M. S. Nawaz, M. Sun, and P. Fournier-Viger. Proof guidance in PVS with sequential pattern mining. In *Proceedings of FSEN 2019*, volume 11761 of *LNCS*, pages 45–60. Springer, 2019.

[15] M. Z. Nawaz, O. Hasan, M. S. Nawaz, P. Fournier-Viger, and M. Sun. Proof searching in HOL4 with genetic algorithm. In *Proceedings of SAC 2020*, pages 513–520. ACM, 2020.

[16] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A proof assistant for higher-Order logic*. Springer, 2002.

[17] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. PVS system guide, PVS prover guide, PVS language reference. Technical report, SRI International, November 2001.

[18] J. E. Santo. Curry-howard for sequent calculus at last! In *Proceedings of TLCA 2015*, volume 38 of *LIPIcs*, pages 165–179, 2015.

[19] K. Slind and M. Norrish. A brief overview of HOL4. In *Proceedings of TPHOL 2008*, volume 5170 of *LNCS*, pages 28–32. Springer, 2008.

[20] L. A. Yang, J. P. Liu, C. H. Chen, and Y. P. Chen. Automatically proving mathematical theorems with evolutionary algorithms and proof assistants. In *Proceddings of CEC 2016*, pages 4421–4428. IEEE, 2016.

[21] X. Yang. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, pages 65–74, 2010.

Formal specification and model checking of Saber lattice-based key encapsulation mechanism in Maude

Duong Dinh Tran*, Kazuhiro Ogata*, Santiago Escobar†, Sedat Akleylek‡ and Ayoub Otmani§
*Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan
Email: {duongtd, ogata}@jaist.ac.jp
†VRAIN, Universitat Politècnica de València, Valencia, Spain
Email: sescobar@upv.es
‡Ondokuz Mayis University, Samsun, Turkey
Email: sedat.akleylek@bil.omu.edu.tr
§University of Rouen Normandie, France
Email: ayoub.otmani@univ-rouen.fr

***Abstract*—The security of most public-key cryptosystems currently in use today is threatened by advances in quantum computing. That is the reason why recently many researchers and industrial companies have spent lots of effort on constructing post-quantum cryptosystems, which are resistant to quantum attackers. A large number of post-quantum key encapsulation mechanisms (KEMs) have been proposed to provide secure key establishment - one of the most important building blocks in asymmetric cryptography. This paper presents a formal security analysis of Saber lattice-based KEM. We first formally specify the mechanism in Maude, a rewriting logic-based specification/programming language equipped with many functionalities, such as a reachability analyzer (or the search command) that can be used as an invariant model checker, and then conduct invariant model checking with the Maude search command, finding an attack.**

***Keywords*-KEM; Maude; post-quantum cryptography; lattice-based cryptography; model checking.**

I. Introduction

The most popular asymmetric (or public-key) primitives used today will become insecure under sufficient strong quantum computers running Shor's algorithm [1]. This is because the hard mathematical problems on which asymmetric primitives rely are hard only under conventional computers, but can be efficiently solved by a sufficient large-scale quantum computer. As a response to the quantum attack threat, there is extensive research to find new schemes which are secure even in the presence of quantum adversaries. In the past few years, many post-quantum asymmetric primitives have been proposed as replacements for those traditional ones currently in use. The National Institute of Standards and Technology of USA (NIST) also started the Post-Quantum Cryptography Project in 2017, calling for proposals of post-quantum cryptographic protocols that are secure against both conventional and quantum computers[1]. There were 82 submissions to this standardization project, implying the importance of this problem. Among these submissions, there are large numbers of proposals for post-quantum key encapsulation mechanisms (KEMs), which aim to securely establish a symmetric key between two parties. This is understandable because the key exchange algorithm can be said to be the most important building block of cryptosystems.

Security analysis of cryptographic primitives and/or protocols can be fundamentally divided into two approaches: computational security and symbolic security. Proof in the computational model requires a definition of secure cryptographic construction (primitive, protocol), and some assumptions about the computationally hard problem. The proof can be regarded as a mathematical reduction, such that it makes sure that the only chance to violate the security of such a construction is to solve the infeasible problem. However, computational proofs are often not easy to understand for non-experts in cryptography. On the other hand, symbolic analysis is easier to understand, computer-verified, and suitable for automation. Our approach presented in this paper belongs to the latter.

We formally specify and model check Saber KEM [2]. In addition, Kyber [3] (precisely CRYSTALS-Kyber) and SK-MLWR [4] (the KEM proposed in [4] is called SK-MLWR in the present paper) are also tackled, but because of space limitation, they are not presented in this paper. We provide the specifications of them at https://github.com/duongtd23/kems-mc. Saber is a KEM whose security is based on the hardness of the Module Learning With Rounding (MLWR)

D. D. Tran and K. Ogata have been supported by JST SICORP Grant Number JPMJSC20C2, Japan.
S. Akleylek has been partially supported by TUBITAK under Grant No.121R006.
S. Escobar has been partially supported by the grant RTI2018-094403-B-C32 funded by MCIN/AEI/10.13039/501100011033 and ERDF A way of making Europe, by the grant PROMETEO/2019/098 funded by Generalitat Valenciana, and by the grant PCI2020-120708-2 funded by MICIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR.
DOI reference number: 10.18293/SEKE2022-097

[1]https://csrc.nist.gov/projects/post-quantum-cryptography

problem, which belongs to lattice-based cryptography. We use Maude [5], a programming/specification language based on rewriting logic, to first formally specify the Dolev-Yao generic intruder [6] as well as these KEMs. By employing the Maude search command, a Man-In-The-Middle (MITM) attack is found. Although this kind of attack is not a novel attack for KEMs, the formal specifications in Maude and the model checking experiments are worth reporting. Our ultimate goal is to come up with a new security analysis/verification technique for post-quantum cryptographic protocols, which use quantum secure primitives, such as Saber. Formally specifying such primitives is necessary for analyzing the security later on. What is described in the paper is our initial step toward the goal.

Related work. In 2012, Blanchet [7] has surveyed various approaches to security protocol verification in both symbolic model and computational model. In the symbolic model, there is a large number of tools existing for verifying cryptosystems, such as ProVerif [8], Maude-NPA [9], and Tamarin [10]. The symbolic protocol verifier ProVerif, which was developed by Blanchet, can automatically prove security properties of cryptographic protocol specifications. ProVerif is based on an abstract representation of the protocol by a set of Horn clauses, and it determines whether the desired security properties hold by resolution on these clauses. The practicability of ProVerif has been demonstrated through case studies, such as [11]. ProVerif can handle an unbounded number of sessions (executions) of protocols, but termination is not guaranteed in general because the resolution algorithm may not terminate. To mitigate this challenge, Escobar et al. [12] proposed some techniques to reduce the size of the search space in Maude-NPA, such as generating formal grammars representing terms (states information) unreachable from initial states and using super lazy intruder to delay the generation of substitution instances as much as possible. Even though, the termination of the tool is not always guaranteed. Among many case studies that demonstrated the capabilities of Maude-NPA, [13] presented one case study with Diffie-Hellman key agreement protocol. Tamarin [10] is another tool for symbolic security verification of cryptographic protocols. Tamarin provides two ways of constructing proofs: fully automated mode and interactive mode. The tool may not terminate in the fully automated mode. In the interactive mode, the tool allows users to provide lemmas that must be proved.

In the computational security approach, game-based model is known as a standard model for proving security. Security for cryptographic primitives or protocols is defined as an attack game played between an *adversary* and some benign entity, which is called the *challenger*. The security proof typically leads to a proof that any supposed adversary can get an advantage over the challenger if and only if he/she is able to solve some computationally hard problem (e.g., discrete logarithm, integer factorization). CryptoVerif [14] is a tool for mechanizing proof in the computational model. It can generate proofs by sequences of games automatically or with little user interaction. Alwen et al. [15] have employed CryptoVerif to analyze the security of the Hybrid Public Key Encryption (HPKE), which is a candidate for a new public key encryption standard.

II. Saber Key Encapsulation Mechanism

A. Key encapsulation mechanism

A key encapsulation mechanism is a tuple of algorithms (KeyGen, Encaps, Decaps) along with a finite keyspace $\mathcal{K}$:

- $\text{KeyGen}() \rightarrow (pk, sk)$: A probabilistic *key generation* algorithm that outputs a public key pk and a secret key sk.
- $\text{Encaps}(pk) \rightarrow (c, k)$: A probabilistic *encapsulation* algorithm that takes as input a public key pk, and outputs a ciphertext (or encapsulated message) c and a key $k \in \mathcal{K}$.
- $\text{Decaps}(c, sk) \rightarrow k$: A (usually deterministic) *decapsulation* algorithm that takes as input a ciphertext c and a secret key sk, and outputs a key $k \in \mathcal{K}$.

A KEM is ϵ-*correct* if for all $(pk, sk) \leftarrow \text{KeyGen}()$ and $(c, k) \leftarrow \text{Encaps}(pk)$, it holds that $\Pr[\text{Decaps}(c, sk) \neq k] \leq \epsilon$. We say it is *correct* if $\epsilon = 0$.

B. Saber

Let $\mathbb{Z}_q$ denote the ring of integers modulo q. Let R and R_q denote the polynomial ring $\mathbb{Z}[X]/(X^n + 1)$ and the quotient polynomial ring $\mathbb{Z}_q[X]/(X^n + 1)$, respectively. Single polynomials are written without markup, bold lower-case letters represent vectors with coefficients in R or R_q, and bold upper-case letters denote matrices. If $\mathcal{X}$ is a probability distribution over a set S, then $x \leftarrow \mathcal{X}$ denotes sampling $x \in S$ according to $\mathcal{X}$. $\mathcal{U}$ denotes the uniform distribution and β_μ is a centered binomial distribution with parameter μ (the samples are in $[-\mu/2, \mu/2]$). $\ll$ and $\gg$ are bitwise shift operations, and when they are used with polynomials and matrices, they are applied to each coefficient. $\mathcal{F}, \mathcal{G}, \mathcal{H}$ are hash functions that are used in Saber. gen is a function that generates a pseudorandom matrix $\mathbf{A} \in R_q^{l \times l}$ from a seed $seed_{\mathbf{A}}$.

Fig. 1 describes the three algorithms (KeyGen, Encaps, Decaps) of Saber.KEM. It employs the three algorithms (KeyGen, Enc, Dec) of Saber.PKE, which are shown in Fig. 2. Note that $\mathbf{h}$, h_1, and h_2 are constants; while ϵ_q, ϵ_p, ϵ_T, and μ receive different values on different security levels. The possible values for all of them can be found in [2]. Let us suppose that Alice performs KEM.KeyGen step and sends a public key pk to Bob. Upon receiving pk, Bob randomly chooses a m, performs KEM.Enc step, and sends back to Alice a ciphertext c. Upon receiving c, Alice performs KEM.Dec step, and computes the value of c'. With a very high probability c' is equal to c, implying that m' on

Alice's side equals m on Bob's side with an overwhelming probability. After that, they can derive the same key K.

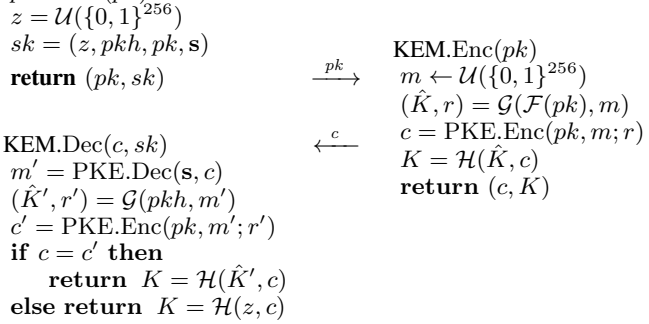


Figure 1. Saber.KEM

PKE.KeyGen()
$seed_{\mathbf{A}} \leftarrow \mathcal{U}(\{0,1\}^{256})$
$\mathbf{A} = \text{gen}(seed_{\mathbf{A}}) \in R_q^{l \times l}$
$r = \mathcal{U}(\{0,1\}^{256})$
$\mathbf{s} = \beta_\mu(R_q^{l \times 1}; r)$
$\mathbf{b} = ((\mathbf{A}^T\mathbf{s} + \mathbf{h}) \bmod q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$
return $(pk := (seed_{\mathbf{A}}, \mathbf{b}), \mathbf{s})$

PKE.Enc($pk = (seed_{\mathbf{A}}, \mathbf{b}), m; r$)
$\mathbf{A} = \text{gen}(seed_{\mathbf{A}}) \in R_q^{l \times l}$
$\mathbf{s}' = \beta_\mu(R_q^{l \times 1}; r)$
$\mathbf{b}' = ((\mathbf{A}\mathbf{s}' + \mathbf{h}) \ mod \ q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$
$v' = \mathbf{b}^T(\mathbf{s}' \bmod p) \in R_p$
$c_m = (v' + h_1 - 2^{\epsilon_p - 1}m \bmod p) \gg (\epsilon_p - \epsilon_T) \in R_T$
return $c := (c_m, \mathbf{b}')$

PKE.Dec($\mathbf{s}, c = (c_m, \mathbf{b}')$)
$v = \mathbf{b}'^T(\mathbf{s} \bmod p) \in R_p$
$m' = ((v - 2^{\epsilon_p - \epsilon_T}c_m + h_2) \bmod p) \gg (\epsilon_p - 1) \in R_2$
return m'

Figure 2. Saber.PKE

III. Formal specification of Saber

A. Formalization of polynomials, vectors, and matrices

We first introduce sort `Poly` that represents polynomials:

```
sort Poly . subsort Int < Poly .
op _p+_ : Poly Poly -> Poly [ctor assoc comm prec 33] .
op _p*_ : Poly Poly -> Poly [ctor assoc comm prec 31] .
op _md_ : Poly Nat -> Poly [ctor prec 32] .
op _p-_ : Poly Poly -> Poly [prec 33] .
op neg_ : Poly -> Poly [ctor] .
```

where `Int` and `Nat` are sorts of integers and natural numbers, respectively. The notation `subsort Int < Poly` indicates that any integer is also a polynomial. `p+`, `p*`, and `p-` denote the addition, multiplication, and subtraction, respectively, between two polynomials. `neg` denotes the negation of a polynomial, while `md` denotes the modulo operation. `assoc comm` indicates that `_p+_` and `_p*_` are declared to be associative and commutative. `prec 33` attached with `_p+_` and `_p-_` indicates that these operators have the same precedence 33, which is lower precedence than that of `_p*_` (i.e., `31`). Let `P1`, `P2`, and `P3` be Maude variables of `Poly`. We define some properties of the operators as follows:

```
eq P1 p+ 0 = P1 .  eq P1 p* 0 = 0 .  eq P1 p* 1 = P1 .
eq P1 p* (P2 p+ P3) = (P1 p* P2) p+ (P1 p* P3) .
eq P1 p+ neg(P1) = 0 .        eq neg(neg(P1)) = P1 .
eq P1 p- P2 = P1 p+ neg(P2) .
eq neg(P1 p+ P2) = neg(P1) p+ neg(P2) .
eq neg(P1 md K) = neg(P1) md K .
```

In a similar way, we introduce sorts `Vector` and `Matrix` representing polynomial vectors and matrices, respectively; operators `v+`, `dot`, and `m*` representing the addition & inner product of two polynomial vectors, and multiplication of a polynomial matrix and a vector, respectively. Let `V1`, `V2`, and `V3` be Maude variables of `Vector`. The declarations of the three operators and the distributive property of vectors are specified as follows:

```
op _v+_ : Vector Vector -> Vector [assoc comm prec 33].
op _dot_ : Vector Vector -> Poly [prec 31] .
op _m*_ : Matrix Vector -> Vector [prec 31] .
eq (V1 v+ V2) dot V3 = (V1 dot V3) p+ (V2 dot V3) .
eq V3 dot (V1 v+ V2) = (V3 dot V1) p+ (V3 dot V2) .
```

B. Formalization of honest parties

Two constructors for the two kinds of messages used in Saber are as follows:

```
op msg1 : Prin Prin Prin PVPair MState -> Msg [ctor] .
op msg2 : Prin Prin Prin PVPair MState -> Msg [ctor] .
```

where `Prin` and `Msg` are sorts denoting principals and messages, respectively. `PVPair` is the sort of polynomial and vector pairs. `MState` is the sort representing message states, receiving one of the following three values: `sent` - the message was sent, `replied` - the message was sent and the receiver replied with another message, and `intercepted` - the message was intercepted by the intruder. The first, second, and third arguments of each of `msg1` and `msg2` are the actual creator, the seeming sender, and the receiver of the corresponding message. The first and last arguments are meta-information that is only available to the outside observer, while the remaining arguments can be seen by every principal.

We model the network as an AC-collection of messages that can be used by the intruder as his/her storage. Consequently, the empty network (i.e., the empty collection) means that no messages have been sent. The intruder can fully control the network, that is he/she can intercept any message, glean information from it, and fake a new message to any honest party. In this paper, a state is expressed as an AC-collection of name-value pairs called observable components. To formally specify Saber in Maude, we use the following observable components:

- (`nw` : $msgs$) - $msgs$ is the AC-collection of messages in the network;

- (keys[p] : $keys$) - $keys$ is an AC-collection of the computed shared keys of principal p. Each entry of $keys$ is in form of key(K,q), where K is the shared key and q is the principal whom p believes that he/she has communicated with;
- (prins : ps) - ps is the collection of all principals participating in the mechanism;
- (seed[p] : sd) - sd is the random seed $seed_{\mathbf{A}}$ (used in Fig. 2) of principal p;
- (r[p] : r_0) - r_0 is the random seed r (used in Fig. 2) of principal p;
- (m[p] : m_0) - m_0 is the random seed m (shown in Fig. 1) of principal p;
- (rd-seed : rds) - rds is a list of available values as the random seed $seed_{\mathbf{A}}$ (we use list, but not set, to reduce the state space for searching). Each time when a principal queries for a random value of $seed_{\mathbf{A}}$, the top value in rds is removed and returned to the principal;
- (rd-r : $rdrs$) - $rdrs$ is a list of the available values as random seed r;
- (rd-m : $rdms$) - $rdms$ is a list of the available values as random seed m;
- (glean-keys : $gkeys$) - $gkeys$ is the AC-collection of shared keys gleaned by the intruder;
- (seeds : sds) - sds is the collection of the random seeds $seed_{\mathrm{A}}$ used by the intruder;
- (rs : rs) - rs is the collection of the random seeds r used by the intruder;
- (ms : ms) - ms is the collection of random seeds m used by the intruder;

Each state in $\mathcal{S}_{Saber}$ is expressed as $\{obs\}$, where obs is an AC-collection of those observable components. We suppose that there are two honest principals alice and bob together with a malicious one, namely eve, participating in Saber.KEM. The initial state init of $\mathcal{I}_{Saber}$ is defined as follows:

```
{(nw: empty) (keys[alice]: empty) (keys[bob]: empty)
(prins: (alice ; bob ; eve)) (rd-seed: (seed1, seed2))
(rd-r: (r1, r2)) (rd-m: (m1, m2)) (glean-keys: empty)
(seed[alice]: 0) (seed[bob]: 0) (m[alice]: 0)
(m[bob]: 0) (seeds: empty) (rs: empty) (ms: empty)}
```

With the honest parties, we specify three transitions: keygen, encaps, and decaps, which correspond to the three steps of the mechanism. We declare Maude constants esp, esq, esT, p, q, h1, and h to denote ϵ_p, ϵ_q, ϵ_T, p, q, h_1, and $\mathbf{h}$, respectively. Let OCs be a Maude variable of observable component collections, A, B, and C be Maude variables of principals (possibly intruder), and PS be a Maude variable of principal collections. Let SD, R, P1, P2, and M, be Maude variables of polynomials; PL and PL2 be Maude variables of polynomial lists. Let G, F, and H denote the hash functions $\mathcal{G}$, $\mathcal{F}$, and $\mathcal{H}$, respectively. Let MS be a Maude variable of networks. The rewrite rule keygen is defined as follows:

```
crl [keygen] : {(rd-seed: (SD, PL)) (rd-r: (R, PL2))
(seed[A]: P1) (r[A]: P2) (prins: (A ; B ; PS))
(nw: MS) OCs}
=> {(rd-seed: PL) (rd-r: PL2)
(seed[A]: SD) (r[A]: R) (prins: (A ; B ; PS))
(nw: (MS ; msg1(A,A,B, pvPair(SD, VB), sent))) OCs}
if MA := gen-A(SD)   /\   S := gen-s(R) /\
VB := shiftRV((tp(MA) m* S v+ h) mdv q, esq - esp) .
```

where MA and VB are Maude variables of polynomial matrices and vectors. tp, shiftRV, and mdv denote matrix transpose, vector bitwise right shift, and vector modulo operations. gen-A and gen-s denote the function gen and the sampling procedures, outputting the matrix $\mathbf{A}$ and the vector $\mathbf{s}$, respectively. The rewrite rule says that when there exist a polynomial SD in rd-seed and a polynomial R in rd-r, A picks it as a random seed r, builds a message msg1 exactly following the KeyGen step, and sends it to B. seed[A] and r[A] are set to SD and R, respectively, and the two values are removed from rd-seed and rd-r.

The rewrite rule encap is defined as follows:

```
crl [encaps] : {(rd-m: (M, PL)) (m[B]: P1) (keys[B]: KS)
(nw: (msg1(C,A,B, pvPair(SD,VB), sent) ; MS)) OCs}
=> {(keys[B]: (KS ; key( H(1st(Kr), CB) , A)))
(nw: (msg1(C,A,B, pvPair(SD, VB), replied) ;
msg2(B,B,A, CB, sent) ; MS)) (rd-m: PL) (m[B]: M)  OCs}
if Kr := G(F(pvPair(SD,VB)), M) /\
   CB := enc(SD,VB,M,2nd(Kr)) .
```

where KS is a Maude variable representing a collection of shared keys, Kr is a Maude variable denoting a pair of polynomials, in which 1st and 2nd are its projection operators. Following the PKE.Enc$(pk, m; r)$ in Fig. 2, enc is defined as follows:

```
ceq enc(SD, VB, M, R) = pvPair(CM, VB')
if MA := gen-A(SD) /\  S' := gen-s(R) /\
VB' := shiftRV((MA m* S' v+ h) mdv q, esq - esp) /\
V' := tpV(VB) dot S' /\
CM := shiftR(((V' p+ h1) p- (2 ^ (esp - 1)) p* M) md p,
       esp - esT) .
```

where tpV(VB) denotes the transpose vector of VB and shiftR denotes the polynomial bitwise right shift. The rewrite rule encaps says that when there exists a message msg1 sent from A to B in the network, B builds a message msg2 exactly following the Encaps step, and sends it back to A. B also computes the shared key with A, and the state of the message msg1 is updated to replied.

The rewrite rule decaps can be defined likewise. Note that we only consider the overwhelming case, i.e., Alice successfully recovers m in Decaps step. We assume that the error tolerance gaps made by error components always be silent, making m' equal m. To this end, we need to define some properties of the bitwise shift operation and polynomials. The first property is as follows: $(2^{\epsilon_p - \epsilon_T} c_m \gg (\epsilon_p - \epsilon_T)) \approx c_m$. It is specified by the following equation:

```
ceq 2PT p* shiftR(CM, PT) = CM
if PT := esp - esT  /\  2PT := 2 ^ PT .
```

where PT and 2PT are variables of integers.

If all coefficients of $\mathbf{s}$ and $\mathbf{s}'$ are small in comparison with p and q, then we have the second property as follows: $(((\mathbf{A}\mathbf{s}' +$

$\mathbf{h}) \bmod q) \gg (\epsilon_q - \epsilon_p))^T \mathbf{s} \approx (((\mathbf{A}^T\mathbf{s} + \mathbf{h}) \bmod q) \gg (\epsilon_q - \epsilon_p))\mathbf{s}'$. The property is specified by the following equation:

```
ceq tpV(shiftRV( (MA m* S' v+ h) mdv q, esq - esp ))
dot S p+ neg(tpV(shiftRV( (tp(MA) m* S v+ h) mdv q,
esq - esp )) dot S') = 0
if isSmall?(S) and isSmall?(S') .
```

where `isSmall?` is a predicate, returning true if all coefficients of `S` (or `S'`) are small in comparison with q. Note that the result of `gen-s` is always defined to be "small," which is done by the following equation: `eq isSmall?(gen-s(R)) = true .`

Finally, to specify the property $((h_2 - h_1 + 2^{\epsilon_p - 1}m) \bmod p) \gg (\epsilon_p - 1) \approx m$, we introduce the following equation:

```
ceq shiftR( (h2 p+ neg(h1) p+ 2P1 p* M) md p, esp - 1)
= M if 2P1 := 2 ^ (esp - 1) .
```

C. Formalization of intruders

We suppose that there is one intruder, namely `eve`, participating in the mechanism. When there exists a message `msg1` sent from `A` to `B` in the network, the intruder can intercept that message, fake a new message, and send it to the receiver. This behavior is specified by the following rewrite rule:

```
crl [keygen-eve] : {(seeds: (SD ; PC1)) (rs: (R ; PC2))
(nw: (msg1(A,A,B, pvPair(SD-A,VB-A), sent) ; MS)) OCs}
=> {(seeds: (SD ; PC1)) (rs: (R ; PC2))
(nw: (msg1(A,A,B, pvPair(SD-A,VB-A), intercepted) ;
msg1(eve,A,B, pvPair(SD, VB), sent) ; MS)) OCs}
if MA := gen-A(SD)   /\   S := gen-s(R) /\
VB := shiftRV((tp(MA) m* S v+ h) mdv q, esq - esp) .
```

where `PC1` and `PC2` are Maude variables of polynomial collections. The intercepted message must have state `sent` at the beginning, which means that the message has not reached the receiver. `eve` then constructs a new faking message from available values `SD` and `R` for the random seeds $seed_{\mathbf{A}}$ and r. These two kinds of random values cannot be gleaned from the network, but `eve` can only construct them by randomly choosing a new value as the rewrite rule `build-ds` as follows:

```
rl [build-sds] : {(rd-seed: (SD, PL)) (seeds: PC1) OCs}
=> {(rd-seed: PL) (seeds: (SD ; PC1)) OCs} .
rl [build-rs] : {(rd-r: (R, PL)) (rs: PC2) OCs}
=> {(rd-r: PL) (rs: (R ; PC2)) OCs} .
```

There are two more rewrite rules `encaps-eve` and `decaps-eve` to specify the intruder's behavior. `encaps-eve` says that when `eve` has intercepted a message `msg1` sent from `A` to `B`, `eve` fakes a new message `msg2`, sends it to `A`, and computes a shared secret key with `A`. `decaps-eve` says that when `eve` has faked a new message `msg1`, sent it to `B`, and `B` on his/her belief that the message truly comes from `A` has replied to `A` a message `msg2`, `eve` intercepts the message `msg2`, and computes a shared secret key with `B`.

IV. Model checking and Man-In-The-Middle-Attack

We introduce the following search command:

Alice.Step-1
$(seed_{\mathbf{A}}, \mathbf{b}, \mathbf{s}) = \mathrm{PKE.KeyGen}()$
$pk = (seed_{\mathbf{A}}, \mathbf{b})$
$pkh = \mathcal{F}(pk)$
return $(pk, sk := (phk, pk, \mathbf{s}))$

Eve.Step-2$(pk = (seed_{\mathbf{A}}, \mathbf{b}))$
$(seed_{\mathbf{A}e}, \mathbf{b}_e, \mathbf{s}_e) = \mathrm{PKE.KeyGen}()$
$pk_e = (seed_{\mathbf{A}e}, \mathbf{b}_e)$
$pkh_e = \mathcal{F}(pk_e)$
return $(pk_e, sk_e := (phk_e, pk_e, \mathbf{s}_e))$

Bob.Step-3$(pk_e = (seed_{\mathbf{A}e}, \mathbf{b}_e))$
$m \leftarrow \mathcal{U}(\{0,1\}^{256})$
$(\hat{K}, r) = \mathcal{G}(\mathcal{F}(pk_e), m)$
$c = \mathrm{PKE.Enc}(pk_e, m; r)$
return $(c, K_b := \mathcal{H}(\hat{K}, c))$

Eve.Step-4$(pk = (seed_{\mathbf{A}}, \mathbf{b}))$
$m_e \leftarrow \mathcal{U}(\{0,1\}^{256})$
$(\hat{K}_e, r_e) = \mathcal{G}(\mathcal{F}(pk), m_e)$
$c_e = \mathrm{PKE.Enc}(pk, m_e; r_e)$
return $(c_e, K_a := \mathcal{H}(\hat{K}_e, c_e))$

Alice.Step-5$(c_e, sk := (phk, pk, \mathbf{s}))$
$m' = \mathrm{PKE.Dec}(\mathbf{s}, c_e)$
$(\hat{K}', r') = \mathcal{G}(pkh, m')$
$c' = \mathrm{PKE.Enc}(pk, m'; r')$
if $c' = c_e$ **then return** $K_a := \mathcal{H}(\hat{K}', c')$

Eve.Step-6$(c, sk_e := (phk_e, pk_e, \mathbf{s}_e))$
$m'_e = \mathrm{PKE.Dec}(\mathbf{s}_e, c)$
$(\hat{K}'_e, r'_e) = \mathcal{G}(pkh_e, m'_e)$
$c'_e = \mathrm{PKE.Enc}(pk_e, m'_e; r'_e)$
if $c = c'_e$ **then return** $K_b := \mathcal{H}(\hat{K}'_e, c)$

Figure 3. A counterexample found by Maude

```
search [1] in Saber : init =>*
{(keys[alice]: key(K1,bob)) (keys[bob]: key(K2,alice))
(glean-keys: (key(K1,alice) key(K2,bob) KS)) OCs} .
```

where `K1` and `K2` are Maude variables that denote arbitrary shared keys. `K1` may or may not equal `K2`. The command tries to find a state reachable from `init` such that: `alice` in her belief obtains the shared key `K1` with `bob`, `bob` in his belief obtains the shared key `K2` with `alice`, and `eve` owns both `K1` and `K2`. Maude found a counterexample, and this kind of vulnerability belongs to MITM attacks. Fig. 3 shows how this attack happens on Saber, which is visualized from the path leading to the counterexample Maude returned. There are mainly six steps as follows:

Step-1: Alice wants to construct a shared key with Bob. She starts by performing KEM.KeyGen, generating a public key pk and a secret key sk. She keeps sk, and send pk to Bob.

Step-2: Eve intercepts the first message sent from Alice to Bob. She follows the KEM.KeyGen step to generate a pair (pk_e, sk_e), impersonating Alice to send pk_e to Bob.

Step-3: Bob receives pk_e thinking it is from Alice. As a response, he takes a random m, performs KEM.Enc with the input pk_e, and obtains a ciphertext c and a shared key K_b. He sends the ciphertext c back to Alice, and keeps the

key K_b, which he believes that it is the shared key obtained by him and Alice.

Step-4: Eve intercepts the replied message which contains the ciphertext c sent from Bob to Alice. Then, she takes a random m_e, performs PKE.Enc with inputs pk and m_e, and obtains a ciphertext c_e and a shared key K_a. She sends the ciphertext c_e back to Alice as a response for the first message.

Step-5: Alice receives the ciphertext c_e thinking it is from Bob. She performs KEM.Dec with inputs c_e and sk, obtains the shared key K_a. She believes that K_a is the shared key obtained by her and Bob.

Step-6: Eve performs KEM.Dec with inputs c and sk_e, and obtains the shared key K_b.

The reachable state space in the experiment is finite. Indeed, if we try to run the following command: `search in SABER : init =>* {OCs} .`, the number of returned solutions is finite, implying that the state space is finite. This can be understandable. The key point is that the numbers of possible values that each observable component (i.e., a name-value pair) can receive are finite.

Remark. Readers may argue that this kind of attack is not a novel attack because Saber KEM does not go along with any solution for authentication. We agree on it. The paper instead illustrates one symbolic approach for reasoning KEMs rather than focusing on this kind of attack. Our ultimate goal is to come up with a new security analysis/verification technique for post-quantum cryptographic protocols, such as quantum-resistant TLS. Such protocols use post-quantum cryptographic primitives, such as KEMs. Formally specifying such primitives is necessary to analyze the security. What is described in the paper is our initial step toward the goal.

V. Conclusion

The paper has presented an approach to security analysis of Saber.KEM in the symbolic model. We first used Maude as a specification language to formally specify the mechanism. After that, by employing Maude search command, an MITM attack was found. The occurrence of the attack is basically because a KEM alone does not come with an authentication solution.

Amazon Web Service team has proposed a post-quantum TLS protocol [16] that uses a hybrid key exchange method: a traditional key exchange algorithm together with a post-quantum KEM. The reason why a post-quantum KEM is required is clear. However, why do we still need to employ a traditional key exchange algorithm? One reason is that most post-quantum KEMs are not studied/analyzed deeply, and thus, nothing guarantees that there is not any potential flaw in them. Thus, deep security analysis of such KEMs in particular and other post-quantum cryptographic primitives/protocols is an important challenge to guarantee their reliability. One piece of our future work is to formally verify the security of the post-quantum TLS protocol against both classical and quantum computers. To this end, the most important task is to come up with a new intruder model because intruders will be able to utilize quantum computers on which quantum algorithms, such as Shor's one [1], run in the post-quantum era.

References

[1] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[2] J.-P. D'Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, "Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM," in *AFRICACRYPT 2018*, 2018, pp. 282–305.

[3] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - kyber: A CCA-Secure Module-Lattice-Based KEM," in *2018 IEEE EuroS&P*, 2018, pp. 353–367.

[4] S. Akleylek and K. Seyhan, "Module learning with rounding based key agreement scheme with modified reconciliation," *Computer Standards & Interfaces*, vol. 79, p. 103549, 2022.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, Eds., *All About Maude*, 2007, vol. 4350.

[6] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[7] B. Blanchet, "Security protocol verification: Symbolic and computational models," in *POST 2012, ETAPS 2012*, vol. 7215. Springer, 2012, pp. 3–29.

[8] ——, "Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif," in *FOSAD 2012/2013 Tutorial Lectures*, vol. 8604, 2013, pp. 54–87.

[9] S. Escobar, C. Meadows, and J. Meseguer, "A Rewriting-Based Inference System for the NRL Protocol Analyzer and Its Meta-Logical Properties," *Theor. Comput. Sci.*, vol. 367, no. 1, p. 162–202, Nov. 2006.

[10] B. Schmidt, S. Meier, C. Cremers, and D. A. Basin, "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties," in *IEEE CSF 2012*, 2012, pp. 78–94.

[11] R. Küsters and T. Truderung, "Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation," in *IEEE CSF 2009*, 2009, pp. 157–171.

[12] S. Escobar, C. A. Meadows, and J. Meseguer, "State Space Reduction in the Maude-NRL Protocol Analyzer," in *ESORICS 2008*, vol. 5283, 2008, pp. 548–562.

[13] S. Escobar, J. Hendrix, C. A. Meadows, and J. Meseguer, "Diffie-Hellman Cryptographic Reasoning in the Maude-NRL Protocol Analyzer," in *Proceeding 2nd International Workshop on Security and Rewriting Techniques*, 2006.

[14] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Secur. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.

[15] J. Alwen, B. Blanchet, E. Hauck, E. Kiltz, B. Lipp, and D. Riepel, "Analysing the HPKE standard," in *EUROCRYPT 2021*, vol. 12696, 2021, pp. 87–116.

[16] M. Campagna and E. Crockett, "Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)," RFC Editor, RFC, 09 2021.

A divide & conquer approach to until and until stable model checking

Canh Minh Do, Yati Phyo, and Kazuhiro Ogata
School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
Nomi, Ishikawa 923-1211 Japan
{canhdominh,yatiphyo,ogata}@jaist.ac.jp

***Abstract*—The paper describes a technique to mitigate the notorious state space explosion in model checking. The technique is called a divide & conquer approach to until and until stable model checking. As indicated by the name, the technique is dedicated to until and until stable properties that are expressed as $\varphi_1 \mathcal{U} \varphi_2$ and $\varphi_1 \mathcal{U} \Box\varphi_2$, respectively, where φ_1, φ_2 are state propositions. For real-time system analysis, some interesting systems requirements are expressed as until and until stable properties. For example, a clock is running and shows a correct time until a certain time has passed or until the clock stops due to an empty battery or other failures. Therefore, it is worth focusing on the properties. For each property, we prove a theorem that the proposed technique is correct and design an algorithm based on the theorem to support the technique.**

***Index Terms*—until properties, until stable properties, linear temporal logic (LTL), model checking.**

I. Introduction

Model checking [1] is an automatic verification technique for verifying finite-state hardware and software systems. It has proven to be a tremendously successful technique to verify requirements for a variety of systems. However, there are still some challenges to tackle in model checking, one of them is the state space explosion, the most annoying one. Many techniques are devised to mitigate the problem, such as ordered binary decision diagrams [2], partial order reduction [3], and abstraction [4]–[6]. Such techniques mitigate the problem to some extent, but the problem still remains and often prevents some model checking experiments from being carried out.

To address the problem, our research group came up with a divide & conquer approach to model checking leads-to properties [7] expressed as $\varphi_1 \rightsquigarrow \varphi_2$, conditional stable properties [8] expressed as $\varphi_1 \rightsquigarrow \Box\varphi_2$, and eventual (or eventually) properties [9] expressed as $\Diamond\varphi$, where $\varphi, \varphi_1, \varphi_2$ are state propositions. Although leads-to properties, conditional stable properties, and eventual properties can be expressed in LTL, it is necessary to individually prove the correctness of each of the three divide & conquer approaches to leads-to, eventual, and conditional stable model checking, come up with each algorithm. This is because it is not straightforward to uniformly deal with the three different properties so as to mitigate the state space explosion. Likewise, it is not straightforward to deal with until and until stable properties that are expressed $\varphi_1 \mathcal{U} \varphi_2$ and $\varphi_1 \mathcal{U} \Box\varphi_2$, respectively, where φ_1, φ_2 are state propositions. Hence, it is meaningful to prove the correctness of the divide & conquer approach to until and until stable model checking and design each algorithm for each property so as to mitigate the state space explosion.

Real-Time Maude (RT-Maude) [10] is a language and tool supporting the formal specification and analysis of real-time and hybrid systems where time information is taken into account. RT-Maude is implemented in Maude [11] as an extension of Full Maude [12]. RT-Maude specifications are executable and simulate the progress in systems under analysis by *timed rewriting*. *Tick rules* are used to formalize the time advance in systems in which the time increment is given in the form of either a concrete value or a variable. The former is called *time-deterministic* and is used in discrete time domains, while the latter is called *time-nondeterministic* because the time increment is an arbitrary number that will be decided based on the *time sampling strategy* (time mode) specified by users among some time sampling strategies supported by RT-Maude, and is used in dense time domains. RT-Maude supports time-bounded linear temporal logic model checking that can analyze all behaviors of a system from a given initial state up to a certain duration. By restricting model checking up to a certain duration, the set of reachable states is a finite set so that model checking experiments can be carried out. RT-Maude usually uses some properties specified in LTL to express systems requirements among which until and until stable properties are used [10], [13]. Besides, RT-Maude supports dedicated commands to check for until and until stable properties, meaning that until and until stable properties are interesting properties in real-time systems. Furthermore, the reachable state space of a real-time system is often huge because the behavior of the system changes over time. Therefore, it is worth focusing on mitigating the state space explosion in until and until stable model checking.

This paper describes an ongoing work that extends the divide & conquer approach so as to handle until and until stable properties expressed as $\varphi_1 \mathcal{U} \varphi_2$ and $\varphi_1 \mathcal{U} \Box\varphi_2$, respectively, where φ_1, φ_2 are state propositions. Until properties informally say that the first argument is true *until* its second argument is true, which is required to happen. Until stable properties informally say that the first argument is true *until*

This research was partially supported by JSPS KAKENHI Grant Number JP19H04082.
DOI reference number: 10.18293/SEKE2022-058

its second argument is true and continues to be true (*stable*) subsequently, which is required to happen. We can see that if until stable properties hold, then until properties also hold. In this paper, for each property, we prove a theorem that the proposed technique is correct and design an algorithm based on the theorem to support the technique. A basic idea of the technique is that the reachable state space from each initial state is split into multiple layers, generating multiple sub-state spaces, and conducting model checking experiments for each sub-state space. If the size of each sub-state space is much smaller than the one of the original reachable state space, it is feasible to conduct model checking experiments with the approach even though it is infeasible to do so for the original reachable state space due to the state space explosion. That is the key to mitigating the state space explosion in model checking with the technique.

The rest of the paper is organized as follows. Sect. II mentions some preliminaries. Sect. III proves a theorem for the divide & conquer approach to until model checking. Sect. IV describes an algorithm that is constructed based on the theorem. Sect. V proves a theorem for the divide & conquer approach to until stable model checking. Sect. VI describes an algorithm that is constructed based on the theorem. Sect. VII mentions some existing related work. Sect. VIII finally concludes the paper together with some future directions.

II. Preliminaries

Definition 1 (Kripke structures)**.** *A Kripke structure* $\boldsymbol{K} \triangleq \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{L} \rangle$ *consists of a set* $\boldsymbol{S}$ *of states, a set* $\boldsymbol{I} \subseteq \boldsymbol{S}$ *of initial states, a left-total binary relation* $\boldsymbol{T} \subseteq \boldsymbol{S} \times \boldsymbol{S}$ *over states, a set* $\boldsymbol{A}$ *of atomic propositions and a labeling function* $\boldsymbol{L}$ *whose type is* $\boldsymbol{S} \to 2^{\boldsymbol{A}}$*. An element* $(s, s') \in \boldsymbol{T}$ *is called a (state) transition from* s *to* s' *and may be written as* $s \to_{\boldsymbol{K}} s'$*.*

An infinite sequence $s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$ of states is called a path of $\boldsymbol{K}$ iff for any natural number i, $(s_i, s_{i+1}) \in \boldsymbol{T}$. Let π be $s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$ and some notations on π are defined as follows: $\pi(i)$ is s_i; π^i is $s_i, s_{i+1}, \ldots$; π_i is $s_0, s_1, \ldots, s_i, s_i, s_i, \ldots$; $\pi^{(i,j)}$ is $s_i, s_{i+1}, \ldots, s_j, s_j, s_j, \ldots$ if $i \leq j$ and $s_i, s_i, s_i, \ldots$ otherwise; $\pi^{(i,\infty)}$ is π^i, where i, j are natural numbers. A path π of $\boldsymbol{K}$ is called a computation of $\boldsymbol{K}$ iff $\pi(0) \in \boldsymbol{I}$. Let $\boldsymbol{P}_{\boldsymbol{K}}$ be the set of all paths of $\boldsymbol{K}$. Let $\boldsymbol{P}_{(\boldsymbol{K},s)}$ be $\{\pi \mid \pi \in \boldsymbol{P}_{\boldsymbol{K}}, \pi(0) = s\}$, where $s \in \boldsymbol{S}$. Let $\boldsymbol{P}^b_{(\boldsymbol{K},s)}$ be $\{\pi_b \mid \pi \in \boldsymbol{P}_{(\boldsymbol{K},s)}\}$, where $s \in \boldsymbol{S}$ and b is a natural number. Note that $\boldsymbol{P}^{\infty}_{(\boldsymbol{K},s)}$ is $\boldsymbol{P}_{(\boldsymbol{K},s)}$.

Definition 2 (Syntax of LTL)**.** *The syntax of linear temporal logic (LTL) is as follows:* $\varphi ::= a \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi$*, where* $a \in \boldsymbol{A}$*.*

Definition 3 (Semantics of LTL)**.** *For any Kripke structure* $\boldsymbol{K}$*, any path* π *of* $\boldsymbol{K}$ *and any LTL formula* φ*,* $\boldsymbol{K}, \pi \models \varphi$ *is inductively defined as follows:*

- $\boldsymbol{K}, \pi \models a$ *iff* $a \in L(\pi(0))$
- $\boldsymbol{K}, \pi \models \top$
- $\boldsymbol{K}, \pi \models \neg\varphi_1$ *iff* $\boldsymbol{K}, \pi \not\models \varphi_1$
- $\boldsymbol{K}, \pi \models \varphi_1 \vee \varphi_2$ *iff* $\boldsymbol{K}, \pi \models \varphi_1$ *and/or* $\boldsymbol{K}, \pi \models \varphi_2$
- $\boldsymbol{K}, \pi \models \bigcirc \varphi_1$ *iff* $\boldsymbol{K}, \pi^1 \models \varphi_1$
- $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2$ *iff there exists a natural number* i *such that* $\boldsymbol{K}, \pi^i \models \varphi_2$ *and for each natural number* $j < i$*,* $\boldsymbol{K}, \pi^j \models \varphi_1$

where φ_1 *and* φ_2 *are LTL formulas. Then,* $\boldsymbol{K} \models \varphi$ *iff* $\boldsymbol{K}, \pi \models \varphi$ *for all computations* π *of* $\boldsymbol{K}$*. Let* $True$ *denote* $\boldsymbol{K}, \pi \models \top$*, which always holds.*

$\bot \triangleq \neg\top$ and some other connectives are defined as follows: $\varphi_1 \wedge \varphi_2 \triangleq \neg((\neg\varphi_1) \vee (\neg\varphi_2))$, $\varphi_1 \Rightarrow \varphi_2 \triangleq (\neg\varphi_1) \vee \varphi_2$, $\varphi_1 \Leftrightarrow \varphi_2 \triangleq (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$, $\Diamond\varphi_1 \triangleq \top \,\mathcal{U}\, \varphi_1$, $\Box\varphi_1 \triangleq \neg(\Diamond\neg\varphi_1)$ and $\varphi_1 \rightsquigarrow \varphi_2 \triangleq \Box(\varphi_1 \Rightarrow \Diamond\varphi_2)$. $\bigcirc$, $\mathcal{U}$, $\Diamond$, $\Box$ and $\rightsquigarrow$ are called next, until, eventually, always and leads-to temporal connectives, respectively. State propositions are LTL formulas such that they do not have any temporal connectives. Until stable properties can be expressed as $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$, where φ_1, φ_2 are state propositions. Although it is unnecessary to define the semantics for $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$, we define it as follows:

- $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2$ iff there exists a natural number i such that $\boldsymbol{K}, \pi^i \models \Box\varphi_2$ and for each natural number $j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$.

III. Multiple Layer Division of Until Model Checking

Proposition 1. *Let* $\boldsymbol{K}$ *be any Kripke structure. If* φ *is any state proposition, then* $(\boldsymbol{K}, \pi \models \varphi) \Leftrightarrow (\boldsymbol{K}, \pi' \models \varphi)$ *for any paths* π *&* π' *of* $\boldsymbol{K}$ *such that* $\pi(0) = \pi'(0)$*.*

Proof. The first state $\pi(0)$ decides if $\boldsymbol{K}, \pi \models \varphi$ holds. □

Lemma 1. *Let* φ_1, φ_2 *be any state propositions of* $\boldsymbol{K}$*. Let* k *be any natural number. Then,* $(\boldsymbol{K}, \pi_k \models \varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow (\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2)$*.*

Proof. If $\boldsymbol{K}, \pi_k \models \varphi_1 \,\mathcal{U}\, \varphi_2$, then there exists $i \leq k$ such that $\boldsymbol{K}, \pi^i_k \models \varphi_2$, which is equivalent to $\boldsymbol{K}, \pi^i \models \varphi_2$ from Proposition 1, and for each $j < i$, $\boldsymbol{K}, \pi^j_k \models \varphi_1$, which is equivalent to $\boldsymbol{K}, \pi^j \models \varphi_1$ from Proposition 1. Hence, $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2$. □

Lemma 2. *Let* φ_1, φ_2 *be any state propositions of* $\boldsymbol{K}$*. Let* k *be any natural number. Then,* $(\boldsymbol{K}, \pi_k \models \Box\varphi_1) \wedge (\boldsymbol{K}, \pi^k \models \varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow (\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2)$*.*

Proof. If $\boldsymbol{K}, \pi_k \models \Box\varphi_1$, then for each $i' \leq k$, $\boldsymbol{K}, \pi^{i'}_k \models \varphi_1$, which is equivalent to $\boldsymbol{K}, \pi^{i'} \models \varphi_1$ from Proposition 1 (1). If $\boldsymbol{K}, \pi^k \models \varphi_1 \,\mathcal{U}\, \varphi_2$ then there exists $i \geq k$ such that $\boldsymbol{K}, \pi^i \models \varphi_2$ and for each j such that $k \leq j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$ (2). From (1) and (2), we have $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2$. □

Lemma 3 (Two layer division of $\varphi_1 \,\mathcal{U}\, \varphi_2$)**.** *Let* φ_1, φ_2 *be any state propositions of* $\boldsymbol{K}$*. Let* k *be any natural number. Then,*

$$
\begin{aligned}
&(\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2) \\
\Leftrightarrow\ &[(\boldsymbol{K}, \pi_k \models \varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow True] \wedge \\
&[(\boldsymbol{K}, \pi_k \not\models \varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow (\boldsymbol{K}, \pi_k \models \Box\varphi_1) \wedge \\
&\qquad\qquad (\boldsymbol{K}, \pi^k \models \varphi_1 \,\mathcal{U}\, \varphi_2)]
\end{aligned}
$$

Proof. (1) Case "only if" (⇒): The case is split into two cases: (1.1) $\boldsymbol{K}, \pi_k \models \varphi_1 \,\mathcal{U}\, \varphi_2$ and (1.2) $\boldsymbol{K}, \pi_k \not\models \varphi_1 \,\mathcal{U}\, \varphi_2$. In

(1.1), it is obvious. In (1.2), from the assumption, there exists i such that $\boldsymbol{K}, \pi^i \models \varphi_2$ and for each $j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$. Because $\boldsymbol{K}, \pi_k \not\models \varphi_1 \mathcal{U} \varphi_2$, then $k < i$. Hence, $(\boldsymbol{K}, \pi_k \models \Box\varphi_1) \wedge (\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \varphi_2)$.

(2) Case "if" ($\Leftarrow$): The case is split into two cases: (2.1) $\boldsymbol{K}, \pi_k \models \varphi_1 \mathcal{U} \varphi_2$ and (2.2) $\boldsymbol{K}, \pi_k \not\models \varphi_1 \mathcal{U} \varphi_2$. In (2.1), $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2$ from Lemma 1. In (2.2), $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2$ from Lemma 2. □

Definition 4 (Until_L)**.** *Let L be any non-zero natural number, k be any natural number and d be any function such that $d(0)$ is 0, $d(x)$ is a natural number for $x = 1, \ldots, L$ and $d(L+1)$ is ∞.*

1) $0 \leq k < L - 1$

$$\begin{aligned}&\mathrm{Until}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k)\\ \triangleq\ &[(\boldsymbol{K}, \pi^{(d(k),d(k+1))} \models \varphi_1 \mathcal{U} \varphi_2) \Rightarrow \mathit{True}] \wedge\\ &[(\boldsymbol{K}, \pi^{(d(k),d(k+1))} \not\models \varphi_1 \mathcal{U} \varphi_2)\\ &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d(k),d(k+1))} \models \Box\varphi_1) \wedge\\ &\quad\ \mathrm{Until}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k+1)]\end{aligned}$$

2) $k = L - 1$

$$\begin{aligned}&\mathrm{Until}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k)\\ \triangleq\ &[(\boldsymbol{K}, \pi^{(d(k),d(k+1))} \models \varphi_1 \mathcal{U} \varphi_2) \Rightarrow \mathit{True}] \wedge\\ &[(\boldsymbol{K}, \pi^{(d(k),d(k+1))} \not\models \varphi_1 \mathcal{U} \varphi_2)\\ &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d(k),d(k+1))} \models \Box\varphi_1) \wedge\\ &\quad\ (\boldsymbol{K}, \pi^{(d(k+1),d(k+2))} \models \varphi_1 \mathcal{U} \varphi_2)]\end{aligned}$$

Theorem 1 ($L+1$ layer division of $\varphi_1 \mathcal{U} \varphi_2$)**.** *Let L be any non-zero natural number. Let $d(0)$ be 0, $d(x)$ be any natural number for $x = 1, \ldots, L$ and $d(L+1)$ be ∞. Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$. Then,*

$$(\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2) \Leftrightarrow \mathrm{Until}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

Proof. By induction on L.

- Base case ($L = 1$): It follows from Lemma 3.
- Induction case ($L = l + 1$): We prove the following:

$$(\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2) \Leftrightarrow \mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

Let d_{l+1} be d used in $\mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ such that $d_{l+1}(0) = 0$, $d_{l+1}(i)$ is an arbitrary natural number for $i = 1, \ldots, l+1$ and $d_{l+1}(l+2) = \infty$. The induction hypothesis is as follows:

$$(\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2) \Leftrightarrow \mathrm{Until}_l(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

Let d_l be d used in $\mathrm{Until}_l(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ such that $d_l(0) = 0$, $d_l(i)$ is an arbitrary natural number for $i = 1, \ldots, l$ and $d_l(l+1) = \infty$. Because $d_{l+1}(i)$ is an arbitrary natural number for $i = 1, \ldots, l+1$, we suppose that $d_{l+1}(1) = d_l(1)$ and $d_{l+1}(i+1) = d_l(i)$ for $i = 1, \ldots, l$. Because π is any path of $\boldsymbol{K}$, π can be replaced with $\pi^{d_l(1)}$. If so, we have the following as an instance of the induction hypothesis:

$$(\boldsymbol{K}, \pi^{d_l(1)} \models \varphi_1 \mathcal{U} \varphi_2) \Leftrightarrow \mathrm{Until}_l(\boldsymbol{K}, \pi^{d_l(1)}, \varphi_1, \varphi_2, 0)$$

From Definition 4, $\mathrm{Until}_l(\boldsymbol{K}, \pi^{d_l(1)}, \varphi_1, \varphi_2, 0)$ is $\mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)$ because $d_l(0) = d_{l+1}(0) = 0$, $d_l(1) = d_{l+1}(1)$ and $d_l(i) = d_{l+1}(i+1)$ for $i = 1, \ldots, l$ and $d_l(l+1) = d_{l+1}(l+2) = \infty$. Therefore, the induction hypothesis instance can be rephrased as follows:

$$(\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \varphi_1 \mathcal{U} \varphi_2) \Leftrightarrow \mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)$$

From Definition 4, $\mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ is

$$\begin{aligned}&[(\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \models \varphi_1 \mathcal{U} \varphi_2) \Rightarrow \mathit{True}] \wedge\\ &[(\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \not\models \varphi_1 \mathcal{U} \varphi_2)\\ &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \models \Box\varphi_1) \wedge\\ &\quad\ \mathrm{Until}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)]\end{aligned}$$

which is

$$\begin{aligned}&[(\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \models \varphi_1 \mathcal{U} \varphi_2) \Rightarrow \mathit{True}] \wedge\\ &[(\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \not\models \varphi_1 \mathcal{U} \varphi_2)\\ &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d_{l+1}(0),d_{l+1}(1))} \models \Box\varphi_1) \wedge\\ &\quad\ (\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \varphi_1 \mathcal{U} \varphi_2)]\end{aligned}$$

because of the induction hypothesis instance. From Lemma 3, this is equivalent to $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2$.

□

IV. A Divide & Conquer Approach to Until Model Checking Algorithm

An algorithm can be constructed based on Theorem 1, which is shown as Algorithm 1. For each initial state $s_0 \in \boldsymbol{I}$, unfolding s_0 by using $\boldsymbol{T}$ such that each node except for s_0 has exactly one incoming edge, an infinite tree whose root is s_0 is made. The infinite tree may have multiple copies of some states. Such an infinite tree can be divided into $L+1$ layers, generating multiple sub-state spaces, and conducting model checking experiments for each sub-state space. If the set of reachable states is finite, the number of different states in each layer and each sub-state space is finite. Theorem 1 makes it possible to check $\boldsymbol{K} \models \varphi_1 \mathcal{U} \varphi_2$ in a stratified way in that for each layer $l \in \{1, \ldots, L+1\}$, we can check $\boldsymbol{K}, s, d(l) \models \varphi$ for each $s \in \{\pi(d(l-1)) \mid \pi \in \boldsymbol{P}^{d(l-1)}_{(\boldsymbol{K}, s_0)}\}$, where $d(0)$ is 0, $d(x)$ is a non-zero natural number for $x = 1, \ldots, L$, $d(L+1)$ is ∞, and φ is $\varphi_1 \mathcal{U} \varphi_2$ or $\Box\varphi_1$.

$\boldsymbol{US}$ and $\boldsymbol{US'}$ are variables to which sets of states are set. Initially, $\boldsymbol{US}$ contains all initial states in $\boldsymbol{I}$ at line 1. For each layer $l = 0, 1, \ldots, L$ in the first **forall** loop, we need to do as follows. Firstly, $\boldsymbol{US'}$ that is used to collect states at each layer is set to an empty set at line 3. Secondly, the code fragment at lines 4 – 10 checks $\varphi_1 \mathcal{U} \varphi_2$ for each path that starts with each state in $\boldsymbol{US}$. If the path satisfies the formula, we do not need to take the path into account. Otherwise, we check whether the path satisfies $\Box\varphi_1$ at line 7. If so, the last state of the path is then added to $\boldsymbol{US'}$ at line 8. Otherwise, the path is a counterexample and Algorithm 1 returns Failure. Finally, $\boldsymbol{US'}$ is assigned to $\boldsymbol{US}$ for the next layer at line 11.

Just after the first **forall** loop in Algorithm 1, $\boldsymbol{US}$ contains the set of states located at bottom of the Lth layer by checking $\varphi_1 \mathcal{U} \varphi_2$ and $\Box\varphi_2$ for some paths obtained from intermediate

Algorithm 1: A divide & conquer approach to until model checking

input : $\boldsymbol{K}$ – a Kripke structure
φ_1, φ_2 – state propositions
L – a non-zero natural number
d – a function such that $d(x)$ is a non-zero natural number for $x = 1, \ldots, L$
output: Success ($\boldsymbol{K} \models \varphi_1 \mathcal{U} \varphi_2$) or Failure ($\boldsymbol{K} \not\models \varphi_1 \mathcal{U} \varphi_2$)

```
1  US ← I
2  forall l ∈ {1, ..., L} do
3  |  US' ← {}
4  |  forall s ∈ US do
5  |  |  forall π ∈ P^{d(l)}_{(K,s)} do
6  |  |  |  if K, π ⊭ φ1 U φ2 then
7  |  |  |  |  if K, π ⊨ □φ1 then
8  |  |  |  |  |  US' ← US' ∪ {π(d(l))}
9  |  |  |  |  else
10 |  |  |  |  |  return Failure
11 |  US ← US'
12 forall s ∈ US do
13 |  forall π ∈ P_{(K,s)} do
14 |  |  if K, π ⊭ φ1 U φ2 then
15 |  |  |  return Failure
16 return Success
```

layers (1st to Lth layers). For the final layer $L+1$, we check $\varphi_1 \mathcal{U} \varphi_2$ for each path that starts with each state in $\boldsymbol{US}$ in the code fragment at lines 12 – 15. If there is a path that does not satisfy the formula, Algorithm 1 returns Failure. Otherwise, Algorithm 1 returns Success at the end.

V. Multiple Layer Division of Until Stable Model Checking

Lemma 4. *Let φ be any state proposition of $\boldsymbol{K}$. Let k be any natural number. Then,* $(\boldsymbol{K}, \pi \models \Box\varphi) \Leftrightarrow (\boldsymbol{K}, \pi_k \models \Box\varphi) \wedge (\boldsymbol{K}, \pi^k \models \Box\varphi)$.

Proof. Because φ is a state proposition, whether it holds only depends on the first state of a given path. If $(\boldsymbol{K}, \pi \models \Box\varphi)$, then φ holds for $\pi(i)$ for all i, and vice versa. If $\boldsymbol{K}, \pi_k \models \Box\varphi$ and $\boldsymbol{K}, \pi^k \models \Box\varphi$, then φ holds for $\pi(i)$ for $i = 0, \ldots, k$ and φ holds for $\pi(i)$ for $i = k, \ldots$, respectively, and therefore φ holds for $\pi(i)$ for all i, and vice versa. □

Lemma 5. *Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$.* $(\boldsymbol{K}, \pi \models \Box\varphi_2) \Rightarrow (\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2)$.

Proof. From the assumption, $\boldsymbol{K}, \pi^0 \models \Box\varphi_2$. □

Lemma 6. *Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$. Let k be any natural number. Then,* $(\boldsymbol{K}, \pi_k \models \Box\varphi_1) \wedge (\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \Box\varphi_2) \Rightarrow (\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2)$.

Proof. If $(\boldsymbol{K}, \pi_k \models \Box\varphi_1)$, then for each $i' \leq k$, $\boldsymbol{K}, \pi_k^{i'} \models \varphi_1$, which is equivalent to $\boldsymbol{K}, \pi^{i'} \models \varphi_1$ from Proposition 1 (1). If $\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \Box\varphi_2$, then there exists $i \geq k$ such that $\boldsymbol{K}, \pi^i \models \Box\varphi_2$ and for each j such that $k \leq j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$ (2). From (1) and (2), we have $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2$. □

Lemma 7. *Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$. Let k be any natural number. Then,* $(\boldsymbol{K}, \pi_k \models \varphi_1 \mathcal{U} \Box\varphi_2) \wedge (\boldsymbol{K}, \pi^k \models \Box\varphi_2) \Rightarrow (\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2)$.

Proof. If $\boldsymbol{K}, \pi_k \models \varphi_1 \mathcal{U} \Box\varphi_2$, then there exists $i \leq k$ such that $\boldsymbol{K}, \pi_k^i \models \Box\varphi_2$, which is equivalent to $\boldsymbol{K}, \pi_k^{i'} \models \varphi_2$ for each $i' \geq i$ (note that $\pi_k^{i'} = \pi_k^k$ if $i' \geq k$), which implies $\boldsymbol{K}, \pi^{i'} \models \varphi_2$ for $k \geq i' \geq i$ from proposition 1, and for each $j < i$, $\boldsymbol{K}, \pi_k^j \models \varphi_1$, which is equivalent to $\boldsymbol{K}, \pi^j \models \varphi_1$ from Proposition 1 (1). If $(\boldsymbol{K}, \pi^k \models \Box\varphi_2)$, then $\boldsymbol{K}, \pi^i \models \varphi_2$ for each $i \geq k$ (2). From (1) and (2), we have $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2$. □

Lemma 8 (Two layer division of $\varphi_1 \mathcal{U} \Box\varphi_2$)**.** *Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$. Let k be any natural number. Then,*

$$
\begin{aligned}
&(\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \Box\varphi_2) \\
\Leftrightarrow\ &[(\boldsymbol{K}, \pi_k \models \Box\varphi_1) \Rightarrow (\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \Box\varphi_2)] \wedge \\
&[(\boldsymbol{K}, \pi_k \not\models \Box\varphi_1) \Rightarrow (\boldsymbol{K}, \pi_k \models \varphi_1 \mathcal{U} \Box\varphi_2) \wedge \\
&\qquad\qquad (\boldsymbol{K}, \pi^k \models \Box\varphi_2)]
\end{aligned}
$$

Proof. (1) Case "only if" ($\Rightarrow$): The case is split into two cases: (1.1) $\boldsymbol{K}, \pi_k \models \Box\varphi_1$ and (1.2) $\boldsymbol{K}, \pi_k \not\models \Box\varphi_1$. From the assumption, there exists i such that $\boldsymbol{K}, \pi^i \models \Box\varphi_2$ and for each $j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$. In (1.1), if $k < i$, then $\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \Box\varphi_2$. Otherwise, if $k \geq i, \boldsymbol{K}, \pi^k \models \Box\varphi_2$ from lemma 4. Hence, $\boldsymbol{K}, \pi^k \models \varphi_1 \mathcal{U} \Box\varphi_2$ from lemma 5. In (1.2), from the assumption, $k \geq i$. $\boldsymbol{K}, \pi^k \models \Box\varphi_2$ from lemma 4. We also have $\boldsymbol{K}, \pi_k \models \varphi_1 \mathcal{U} \Box\varphi_2$ because there exists such i in the assumption.

(2) Case "if" ($\Leftarrow$): The case is split into two cases: (2.1) $\boldsymbol{K}, \pi_k \models \Box\varphi_1$ and (2.2) $\boldsymbol{K}, \pi_k \not\models \Box\varphi_1$. In (2.1), $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2$ from Lemma 6. In (2.2), $\boldsymbol{K}, \pi \models \varphi_1 \mathcal{U} \varphi_2$ from Lemma 7. □

Definition 5 ($\mathrm{UStable}_L$)**.** *Let L be any non-zero natural number, k be any natural number and d be any function such that $d(0)$ is 0, $d(x)$ is a natural number for $x = 1, \ldots, L$ and $d(L+1)$ is ∞.*

1) $0 \leq k < L - 1$

$$
\begin{aligned}
&\mathrm{UStable}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k) \\
&\triangleq\ [(\boldsymbol{K}, \pi^{(d(k), d(k+1))} \models \Box\varphi_1) \\
&\qquad \Rightarrow \mathrm{UStable}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k+1)] \wedge \\
&\quad [(\boldsymbol{K}, \pi^{(d(k), d(k+1))} \not\models \Box\varphi_1) \\
&\qquad \Rightarrow (\boldsymbol{K}, \pi^{(d(k), d(k+1))} \models \varphi_1 \mathcal{U} \Box\varphi_2) \wedge \\
&\qquad\quad (\boldsymbol{K}, \pi^{d(k+1)} \models \Box\varphi_2)]
\end{aligned}
$$

2) $k = L - 1$

$$
\begin{aligned}
&\mathrm{UStable}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, k) \\
&\triangleq\ [(\boldsymbol{K}, \pi^{(d(k), d(k+1))} \models \Box\varphi_1) \\
&\qquad \Rightarrow (\boldsymbol{K}, \pi^{(d(k+1), d(k+2))} \models \varphi_1 \mathcal{U} \Box\varphi_2)] \wedge \\
&\quad [(\boldsymbol{K}, \pi^{(d(k), d(k+1))} \not\models \Box\varphi_1) \\
&\qquad \Rightarrow (\boldsymbol{K}, \pi^{(d(k), d(k+1))} \models \varphi_1 \mathcal{U} \Box\varphi_2) \wedge \\
&\qquad\quad (\boldsymbol{K}, \pi^{d(k+1)} \models \Box\varphi_2)]
\end{aligned}
$$

Theorem 2 ($L+1$ layer division of $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$)**.** *Let L be any non-zero natural number. Let $d(0)$ be 0, $d(x)$ be any natural number for $x = 1, \ldots, L$ and $d(L+1)$ be ∞. Let φ_1, φ_2 be any state propositions of $\boldsymbol{K}$. Then,*

$$(\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \Leftrightarrow \mathrm{UStable}_L(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

Proof. By induction on L.

- Base case ($L = 1$): It follows from Lemma 8.
- Induction case ($L = l + 1$): We prove the following:

  $$(\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \Leftrightarrow \mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

  Let d_{l+1} be d used in $\mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ such that $d_{l+1}(0) = 0$, $d_{l+1}(i)$ is an arbitrary natural number for $i = 1, \ldots, l+1$ and $d_{l+1}(l+2) = \infty$. The induction hypothesis is as follows:

  $$(\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \Leftrightarrow \mathrm{UStable}_l(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$$

  Let d_l be d used in $\mathrm{UStable}_l(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ such that $d_l(0) = 0$, $d_l(i)$ is an arbitrary natural number for $i = 1, \ldots, l$ and $d_l(l+1) = \infty$. Because $d_{l+1}(i)$ is an arbitrary natural number for $i = 1, \ldots, l+1$, we suppose that $d_{l+1}(1) = d_l(1)$ and $d_{l+1}(i+1) = d_l(i)$ for $i = 1, \ldots, l$. Because π is any path of $\boldsymbol{K}$, π can be replaced with $\pi^{d_l(1)}$. If so, we have the following as an instance of the induction hypothesis:

  $$(\boldsymbol{K}, \pi^{d_l(1)} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \Leftrightarrow \mathrm{UStable}_l(\boldsymbol{K}, \pi^{d_l(1)}, \varphi_1, \varphi_2, 0)$$

  From Definition 5, $\mathrm{UStable}_l(\boldsymbol{K}, \pi^{d_l(1)}, \varphi_1, \varphi_2, 0)$ is $\mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)$ because $d_l(0) = d_{l+1}(0) = 0$, $d_l(1) = d_{l+1}(1)$, and $d_l(i) = d_{l+1}(i+1)$ for $i = 1, \ldots, l$, and $d_l(l+1) = d_{l+1}(l+2) = \infty$. Therefore, the induction hypothesis instance can be rephrased as follows:

  $$(\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \Leftrightarrow \mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)$$

  From Definition 5, $\mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 0)$ is

  $$\begin{aligned}
  &[(\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1))} \models \Box\varphi_1) \\
  &\quad \Rightarrow \mathrm{UStable}_{l+1}(\boldsymbol{K}, \pi, \varphi_1, \varphi_2, 1)] \wedge \\
  &[(\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1))} \not\models \Box\varphi_1) \\
  &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1)} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \wedge \\
  &\qquad (\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \Box\varphi_2)]
  \end{aligned}$$

  which is

  $$\begin{aligned}
  &[(\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1))} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \\
  &\quad \Rightarrow (\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2)] \wedge \\
  &[(\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1))} \not\models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \\
  &\quad \Rightarrow (\boldsymbol{K}, \pi^{(d_{l+1}(0), d_{l+1}(1))} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2) \wedge \\
  &\qquad (\boldsymbol{K}, \pi^{d_{l+1}(1)} \models \Box\varphi_2)]
  \end{aligned}$$

  because of the induction hypothesis instance. From Lemma 8, this is equivalent to $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2$.

□

VI. A Divide & Conquer Approach to Until Stable Model Checking Algorithm

An algorithm can be constructed based on Theorem 2, which is shown as Algorithm 2. For each initial state $s_0 \in \boldsymbol{I}$, the reachable state space from s_0 is divided into $L+1$ layers, generating multiple sub-state spaces, and conducting model checking experiments for each sub-state space. Theorem 2 makes it possible to check $\boldsymbol{K} \models \varphi_1 \,\mathcal{U}\, \Box\varphi_2$ in a stratified way in that for each layer $l \in \{1, \ldots, L+1\}$, we can check $\boldsymbol{K}, s, d(l) \models \varphi$ for each $s \in \{\pi(d(l-1)) \mid \pi \in \boldsymbol{P}^{d(l-1)}_{(\boldsymbol{K}, s_0)}\}$, where $d(0)$ is 0, $d(x)$ is a non-zero natural number for $x = 1, \ldots, L$, $d(L+1)$ is ∞, and φ is $\Box\varphi_1$, or $\Box\varphi_2$, or $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$.

$\boldsymbol{CxS}$, $\boldsymbol{CxS'}$, $\boldsymbol{NCxS}$, and $\boldsymbol{NCxS'}$ are variables to which sets of states are set. Initially, $\boldsymbol{CxS}$ contains all initial states in $\boldsymbol{I}$ at line 1 while $\boldsymbol{NCxS}$ is set to an empty set at line 2. For each layer $l = 0, 1, \ldots, L$ in the first **forall** loop, we need to do as follows. Firstly, $\boldsymbol{CxS'}$ and $\boldsymbol{NCxS'}$ that are used to collect states at each layer are set to an empty set at lines 4 and 5, respectively. Secondly, the code fragment at lines 6 – 14 checks $\Box\varphi_1$ for each path that starts with each state in $\boldsymbol{CxS}$. If the path satisfies the formula, the last state of the path is added to $\boldsymbol{CxS'}$ at line 9. Otherwise, we check whether the path satisfies $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$ at line 11. If so, the last state of the path is added to $\boldsymbol{NCxS'}$ at line 12. Otherwise, the path is a counterexample and Algorithm 2 returns Failure at line 14. Thirdly, the code fragment at lines 15 – 20 checks $\Box\varphi_2$ for each path that starts with each state in $\boldsymbol{NCxS}$. If the path satisfies the formula, the last state of the path is added to $\boldsymbol{NCxS'}$ at line 18. Otherwise, the path is a counterexample and Algorithm 2 returns Failure at line 20. Finally, $\boldsymbol{CxS'}$ and $\boldsymbol{NCxS'}$ are assigned to $\boldsymbol{CxS}$ and $\boldsymbol{NCxS}$ for the next layer at lines 21 and 22, respectively.

Just after the first **forall** loop in Algorithm 2, $\boldsymbol{CxS}$ and $\boldsymbol{NCxS}$ contains the sets of states located at bottom of the Lth layer by checking $\Box\varphi_1$, $\varphi_1 \,\mathcal{U}\, \varphi_2$, and $\Box\varphi_2$ for some paths obtained from intermediate layers (1st to Lth layers). For the final layer $L+1$, we check $\varphi_1 \,\mathcal{U}\, \Box\varphi_2$ for each path that starts with each state in $\boldsymbol{CxS}$ in the code fragment at lines 23 – 26. Meanwhile, we check $\Box\varphi_2$ for each path that starts with each state in $\boldsymbol{NCxS}$ in the code fragment at lines 27 – 30. If there is a path that does not satisfy the formula concerned, Algorithm 2 returns Failure. Otherwise, Algorithm 2 returns Success at the end.

VII. Related Work

SAT/SMT-based bounded model checking (BMC) is an effective technique to mitigate the state space explosion problem in model checking. BMC can find a flaw located within some reasonably shallow depth k for each initial state by formalizing the verification problem into an equisatisfiable conjunctive normal form (CNF) formula that can be analyzed by a SAT/SMT solver. An extension of SAT/SMT-based BMC to model check concurrent programs is Lazy Sequentialization (Lazy-CSeq) [14]. Given a concurrent program P together

Algorithm 2: A divide & conquer approach to until stable model checking

input : $\boldsymbol{K}$ – a Kripke structure
φ_1, φ_2 – state propositions
L – a non-zero natural number
d – a function such that $d(x)$ is a non-zero natural number for $x = 1, \ldots, L$
output: Success ($\boldsymbol{K} \models \varphi_1 \ \mathcal{U} \ \Box\varphi_2$) or Failure ($\boldsymbol{K} \not\models \varphi_1 \ \mathcal{U} \ \Box\varphi_2$)

$\boldsymbol{CxS} \leftarrow \boldsymbol{I}$
$\boldsymbol{NCxS} \leftarrow \emptyset$
forall $l \in \{1, \ldots, L\}$ **do**
  $\boldsymbol{CxS'} \leftarrow \{\}$
  $\boldsymbol{NCxS'} \leftarrow \{\}$
  forall $s \in \boldsymbol{CxS}$ **do**
    forall $\pi \in \boldsymbol{P}^{d(l)}_{(\boldsymbol{K},s)}$ **do**
      if $\boldsymbol{K}, \pi \models \Box\varphi_1$ **then**
        $\boldsymbol{CxS'} \leftarrow \boldsymbol{CxS'} \cup \{\pi(d(l))\}$
      else
        if $\boldsymbol{K}, \pi \models \varphi_1 \ \mathcal{U} \ \Box\varphi_2$ **then**
          $\boldsymbol{NCxS'} \leftarrow \boldsymbol{NCxS'} \cup \{\pi(d(l))\}$
        else
          return Failure
  forall $s \in \boldsymbol{NCxS}$ **do**
    forall $\pi \in \boldsymbol{P}^{d(l)}_{(\boldsymbol{K},s)}$ **do**
      if $\boldsymbol{K}, \pi \models \Box\varphi_2$ **then**
        $\boldsymbol{NCxS'} \leftarrow \boldsymbol{NCxS'} \cup \{\pi(d(l))\}$
      else
        return Failure
  $\boldsymbol{CxS} \leftarrow \boldsymbol{CxS'}$
  $\boldsymbol{NCxS} \leftarrow \boldsymbol{NCxS'}$
forall $s \in \boldsymbol{CxS}$ **do**
  forall $\pi \in \boldsymbol{P}_{(\boldsymbol{K},s)}$ **do**
    if $\boldsymbol{K}, \pi \not\models \varphi_1 \ \mathcal{U} \ \Box\varphi_2$ **then**
      return Failure
forall $s \in \boldsymbol{NCxS}$ **do**
  forall $\pi \in \boldsymbol{P}_{(\boldsymbol{K},s)}$ **do**
    if $\boldsymbol{K}, \pi \not\models \Box\varphi_2$ **then**
      return Failure
return Success

with two parameters u and r that are the loop unwinding bound and the number of round-robin schedules, respectively, they first generate an intermediate bounded program P_u by unwinding all loops and inlining all function calls in P with u as a bound except for those used for creating threads. P_u then is transformed into a sequential program $Q_{u,r}$ that simulates all behaviors of P_u within r round-robin schedules. $Q_{u,r}$ is then transformed into a propositional formula that can be analyzed by a SAT/SMT solver. When the size of the system under test is large, the propositional formula becomes complex and the performance of the SAT/SMT solver is degraded or the model checking may become infeasible. To make it possible to conduct model checking experiments. They decompose the set of execution traces of concurrent programs into symbolic subsets [15] so that the single formula is divided into multiple smaller propositional sub-formulas, which then are possibly analyzed by the SAT/SMT solver independently. Their technique is able to deal with safety properties, while our technique is able to deal with until and until stable properties, a class of liveness properties.

VIII. Conclusion

We have described the divide & conquer approach to until and until stable model checking in which for each property, we have proved a theorem that the proposed technique is correct and designed an algorithm based on the theorem to support the technique. As one piece of our future work, we will build a tool supporting the proposed technique and conduct case studies in real-time systems, particularly with RT-Maude, demonstrating that the proposed technique and tool are useful.

References

[1] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds., *Handbook of Model Checking*. Berlin, Heidelberg: Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-10575-8

[2] R. E. Bryant and C. Meinel, *Ordered Binary Decision Diagrams*. Boston, MA: Springer US, 2002, pp. 285–307. [Online]. Available: https://doi.org/10.1007/978-1-4615-0817-5_11

[3] E. M. Clarke, O. Grumberg, M. Minea, and D. A. Peled, "State space reduction using partial order techniques," *Int. J. Softw. Tools Technol. Transf.*, vol. 2, no. 3, pp. 279–287, 1999.

[4] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 5, pp. 1512–1542, 1994.

[5] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *J. ACM*, vol. 50, no. 5, pp. 752–794, 2003.

[6] J. Meseguer, M. Palomino, and N. Martí-Oliet, "Equational abstractions," *Theor. Comput. Sci.*, vol. 403, no. 2-3, pp. 239–264, 2008.

[7] Y. Phyo, C. M. Do, and K. Ogata, "A divide & conquer approach to leads-to model checking," *The Computer Journal*, 2021. [Online]. Available: https://doi.org/10.1093/comjnl/bxaa183

[8] ——, "A divide & conquer approach to conditional stable model checking," in *18th ICTAC*, 2021, pp. 105–111. [Online]. Available: https://doi.org/10.1007/978-3-030-85315-0_7

[9] M. N. Aung, Y. Phyo, C. M. Do, and K. Ogata, "A divide & conquer approach to eventual checking," *Mathematics*, vol. 9, p. 368, 2021. [Online]. Available: https://doi.org/10.3390/math9040368

[10] P. C. Ölveczky, "Real-time maude and its applications," in *Rewriting Logic and Its Applications*, S. Escobar, Ed. Cham: Springer International Publishing, 2014, pp. 42–79.

[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, Eds., *All About Maude*, ser. LNCS. Springer, 2007, vol. 4350.

[12] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, "Full maude: Extending core maude," 01 2007, pp. 559–597.

[13] P. C. Ölveczky, M. Keaton, J. Meseguer, C. L. Talcott, and S. Zabele, "Specification and analysis of the aer/nca active network protocol suite in real-time maude," in *Proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering*, ser. FASE '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 333–348.

[14] O. Inverso, E. Tomasco, B. Fischer, S. La Torre, and G. Parlato, "Bounded verification of multi-threaded programs via lazy sequentialization," *ACM Trans. Program. Lang. Syst.*, vol. 44, no. 1, dec 2021.

[15] O. Inverso and C. Trubiani, "Parallel and distributed bounded model checking of multi-threaded programs," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 202–216.

Managing Risks in Agile Methods: a Systematic Literature Mapping

Fernando Vedoin Garcia
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, Brazil
fevedoingarcia@gmail.com

Jean Carlo R. Hauck
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, Brazil
jean.hauck@ufsc.br

Fernanda N. Rizzo Hahn
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, Brazil
fernandanrizzo@gmail.com

***Abstract*—Agile software development methods have been around since at least 2001. They accommodate changing requirements with the flexibility to deal with cost and scope and have increasingly been used. However, explicit risk management is often ignored as agile methods deal with risk intrinsically and focus on rapid value delivery. In certain contexts, explicit risk management practices are needed to complement agile methods. Thus, this paper presents a systematic literature mapping aiming to discover how do software organizations integrate explicit risk management practices into agile methods. As a result we found 23 primary studies that, in majority, applied case studies in the industry, using agile methods such as Scrum, and adapting agile practices such as Daily Meeting and Iteration Planning Meeting to manage risks related to schedule and communication, for example. The selected primary studies raise evidence that the introduction of explicit risk management practices bring benefits to agile methods.**

***Index Terms*—software, risk management, agile methods, agile practices**

I. Introduction

Agile software development methods [1] have been widely used in software organizations due to their ability to accommodate changing requirements and flexibility to handle cost, scope and software quality according to customer needs [2]. One of the main advantages of adopting agile methods is their ability to reduce risks [3], which leads to successful and timely software development and deployment. Projects that apply agile methods usually make use of frequent reviews in each development cycle and cross-functional project teams to accelerate knowledge sharing and ensure that risks are understood and implicitly managed [4]. The implicit ability to reduce risks has also been one of the main reasons for adopting agile methods in software organizations [5].

However, despite its importance, risk management is often overlooked in agile software development methods as its focus is on rapid value delivery [6]. Even with the adoption of agile methods and investments in software development, failure of software projects is still frequent, increasing the importance of the software development risk management [7].

The explicit application of risk management consists of inserting principles and practices of risk management in the already used practices of lifecycle management [7], thus risks can be identified, analyzed and managed during each software development iteration [4].

Complementing agile methods with explicit risk management practices, has attracted recent interest. Esteki et al. [8], integrates Scrum with the PRINCE2 delivery layer; Schön et al. [9] standardizes risk increasing transparency in the context of multidisciplinary projects; Hayat et al. [10] estimate the impact of risk and convert it into risk detection and control actions. Risk management in software projects has even attracted the application of Machine Learning (ML) aiming to identify or predict risks before project development starts [11].

However, the existing Software Engineering literature lacks insights into the extent to which the combination of agile methods and risk management processes is being applied [6]. Thus, this paper presents a Systematic Literature Mapping (SLM) [12] to answer the research question "How do software organizations integrate explicit risk management practices into agile methods?".

The main contributions of this work are twofold: (i) for Software Engineering researchers we present an extensive survey, to the best of our knowledge, of the state of the art of risk management practices in agile methods; (ii) for practitioners that are seeking to include explicit risk management practices in agile methods we present the most used risk management practices and typical managed risks.

II. Related Works

As primary studies have reported the integration of explicit risk management practices into agile methods, some secondary studies have analyzed this phenomenon from different perspectives.

Vieira, Hauck, and Matalonga [13] conducted an SLM in order to understand how explicit risk management is being integrated into agile software development methods. With 18 selected papers, authors found that the results of integrating explicit risk management with agile methods are positive. The secondary study, however, is not focused on empirical primary studies and not addresses which risk management practices have been applied empirically in real environments.

DOI reference number: 10.18293/SEKE2022-123

Chadli and Idri [14] identified risk mitigation strategies that target Global Software Development (GSD) through a Systematic Literature Review (SLR). The analysis of the 24 selected primary studies resulted in 39 risk factors and 58 mitigation strategies. The strategies were classified by areas such as task-actor, task-structure, and task-technology. The secondary study, however, do not analyze risk management practices nor the specific context of use.

Podari et al. [15] conducted an SLR selecting 52 papers that identify the risks and challenges that affect globally distributed projects and how agile methods can be useful in managing these barriers. The selected primary studies are only focused on GSD, not covering other types of projects.

Thus, it was not possible to find so far in the literature a comprehensive analysis of the introduction of explicit risk management practices in agile methods and the specific practices adopted.

III. Methods

In order to analyze the state of the art of the integration of explicit risk practices in agile methods, we undertook a Systematic Literature Mapping (SLM) following the procedures defined by Petersen, Vakkalanka, and Kuzniarz [12], Petersen et al. [16], and Wohlin [17]. Based on the identified research need, the general research question was defined as: "How do software organizations integrate explicit risk management practices into agile methods?". Thus, we derived the main research question in four detailed analysis questions, as presented in Table I.

TABLE I
Research Questions

	Description
Q1	What are the studies that deal with the integration of risk management practices in agile methods?
Q2	What is the context of use of risk management practices in agile methods?
Q3	What risk management practices are introduced in agile methods?
Q4	What types of risks are managed?

A. Search strategy

The search string was defined following [18], using the most used agile methods [5] and well known terms as synonyms for "agile methods". The search string was then tested and refined by the authors, using previously known primary studies as a reference, resulting in the following search string:

"risk" AND ("agile" OR "scrum" OR "xp" OR "extreme programming" OR "lean" OR "kanban" OR "scrumban" OR "fdd" OR "feature driven development" OR "crystal" OR "iterative development") AND "software"

The search string was applied to the following digital libraries: IEEEXplore, ACM Digital Library, and Scopus, due to their relevance to the software engineering area [19]. The search string was adapted to the specific syntax of each library and applied to title and abstracts fields. The Snowballing technique [17] was also performed using the selected papers from the automated search as input.

Based on the main research question, the following inclusion criteria (IC) and exclusion (EC) criteria were defined: (IC1) Peer reviewed primary studies; (IC2) Written in English; (IC3) Full papers with at least 4 pages; (EC1) Theoretical work/proposal not empirically applied; (EC2) Duplicate studies; (EC3) No full text available; (EC4) Not focused on software development.

B. Study Selection

The selection of studies was performed from July to December of 2021 in four cycles, as presented in Fig. 1.

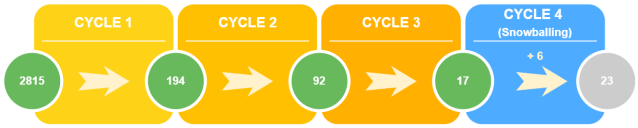


Fig. 1. Number of primary studies by cycle.

In the Cycle 1 the search string was applied to the digital libraries. The resulting list of 2815 primary studies was then divided between the first and third authors, who separately applied the inclusion and exclusion criteria to all paper titles, peer reviewing the results. This initial selection was reviewed by the second author, resulting in 194 selected papers. In Cycle 2, the initial list of papers was filtered by the first and third authors applying the inclusion and exclusion criteria to the papers' summaries, resulting in 92 selected papers, once again reviewed by the second author. In Cycle 3 we merged the lists of papers and filtered the studies on a full-text basis using the inclusion and exclusion criteria, resulting in 17 selected papers after the second author reviewing. Finally, in Cycle 4 the Backward Snowballing technique [17] was applied by the first author using as input the 17 selected papers, resulting in six more papers being selected. After each cycle a meeting was performed with the three authors resolving any possible discordance or inconsistencies. The number of studies for each digital library and cycle is presented in Table II.

TABLE II
Results per Digital Library and Cycle

Digital library	Total	Cycle 1	Cycle 2	Cycle 3
ACM	971	42	15	5
IEEEXplore	957	79	39	5
Scopus	887	73	38	7

IV. Data Collection and Analysis

The 23 selected primary studies are distributed between the years 2000 and 2020. The concentration of works (12) between 2017 and 2020, and the exponential trend line, shown in Fig. 2, indicate the growing relevance of this topic in recent years.

Next, data collected from selected primary studies are presented and analyzed according to each predefined Analysis Question. Extracted raw data is available at: bit.ly/36i7Wby.

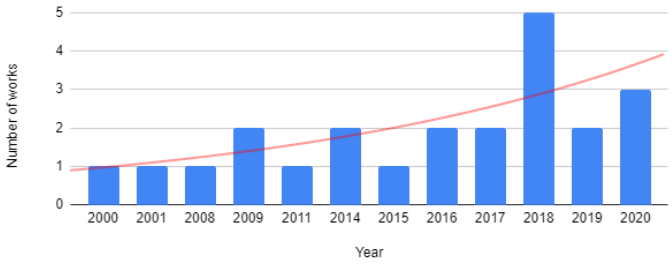


Fig. 2. Distribution of the selected papers per year.

Q1. What are the studies that deal with the integration of risk management practices in agile methods?

The selected primary studies are presented in Table III.

TABLE III
SELECTED STUDIES

#	Title	Ref
S1	Reference Framework and Model for Integration of Risk Management in Agile Systems Engineering Lifecycle of the Defense Acquisition Management Framework.	[20]
S2	A risk management framework for distributed scrum using PRINCE2 methodology.	[8]
S3	A Risk Management Tool for Agile Software Development	[21]
S4	Improving Risk Management in a Scaled Agile Environment	[9]
S5	Risk Assessment Forum	[22]
S6	Agile risk management using software agents	[23]
S7	A risk poker based testing model for scrum	[24]
S8	Agile approach with Kanban in information security risk management	[25]
S9	Integrating Risk Management in Scrum Framework	[26]
S10	Prioritizing and optimizing risk factors in agile software development	[27]
S11	Value-Risk Trade-off Analysis for Iteration Planning in Extreme Programming	[28]
S12	A case study for the implementation of an agile risk management process in multiple projects environments	[29]
S13	A SYSML-Based Approach for Requirements Risk Management and Change Control	[10]
S14	Risk Management for Agile Projects in Offshore Vietnam	[30]
S15	An industrial case study of implementing software risk management	[31]
S16	Characterization of risky projects based on project managers' evaluation	[32]
S17	Characterization and prediction of issue-related risks in software projects	[33]
S18	Outlining a Model Integrating Risk Management and Agile Software Development	[34]
S19	Lightweight Risk Management in Agile Projects	[35]
S20	A risk management framework for distributed agile projects	[36]
S21	Implementation of Risk Management with SCRUM to Achieve CMMI Requirements	[37]
S22	A New Project Risk Management Model based on Scrum Framework and Prince2 Methodology	[38]
S23	Risks to Effective Knowledge Sharing in Agile Software Teams: A Model for Assessing and Mitigating Risks	[39]

Q2. What is the context of use of risk management practices in agile methods?

We define the context of use as: (Q2.1) the type of application environment, (Q2.2) agile method adopted, (Q2.3) type of empirical study and (2.4) number of organizations involved. The context-related data is summarized in Table IV.

The selected studies were applied in two different environments (Q2.1): software industry or academia. 18 (78%) studies were applied in software development organizations and 5 studies (22%) in an academic environment.

Regarding the agile methods adopted (Q2.2), 14 (61%) adopted Scrum, 3 (13%) adopted XP, 2 (9%) cited Kanban, and only 1 (4%) mentioned the Dynamic System Development Method (DSDM), whereas it is not explicit in the search string. Among the selected studies, 7 (30%) did not mention any specific agile method. The total is greater than 100%, as some studies used more than one agile method.

In the industry environment, the agile methods that appeared the most were Scrum (10 - 43%) and XP (3 - 13%). In academia, the predominant method was also Scrum (4 - 17%). Study S3 was the only study applied in academy environment that did not mention any specific agile method.

As for the type of empirical study (Q2.3), 17 (74%) applied case studies, 2 (9%) applied experiments, 3 (13%) applied surveys, and only S8 (4%) applied a proof of concept. It is possible to observe that in the industry most applications were case studies, while in academia there was a balance. The approach proposed in S6 was validated with two case studies.

Most (15 - 65%) of the studies were applied in only 1 organization (Q2.4). Study S17, in turn, was applied in 5 organizations with projects that differ significantly in size, complexity, development process, and community size.

TABLE IV
CONTEXT OF USE

Question	Extracted data	
Q2.1 - Context	Industry	S1, S2, S4, S5, S8, S10, S11, S12, S13, S14, S15, S16, S17, S18, S20, S21, S22, S23
	Academy	S3, S6, S7, S9, S19
Q2.2 - Agile method	Scrum	S2, S5, S6, S7, S9, S10, S12, S14, S18, S19, S20, S21, S22, S23
	XP	S10, S11, S14
	Kanban	S8, S14
	DSDM	S10
	Undefined	S1, S3, S4, S13, S15, S16, S17
Q2.3 - Type	Case study	S1, S2, S4, S5, S6, S7, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20
	Experiment	S3, S9
	Concept proof	S8
	Survey	S21, S22, S23
Q2.4 - Instances	Exactly 1	S1, S2, S4, S5, S6, S7, S8, S11, S12, S13, S14, S15, S16, S18, S19
	Between 2 and 10	S10, S17, S20, S23
	Undefined	S3, S9, S21, S22

Q3. What risk management practices are introduced in agile methods?

Two different strategies were adopted by the studies to integrate risk management into agile methods: using existing agile practices or introduce new risk management practices.

Studies S3 and S9 adopted the first strategy. The most commonly used practices are Brainstorming, Pair Programming, Daily Meetings, Incremental Deliveries and Prototyping.

Adopting the second strategy, the other primary studies have created new agile practices or introduced adapted traditional practices into agile methods to improve risk management. Table V presents the introduced risk management practices, grouped by the existing agile practice impacted (when applicable). Some examples of practices are described below.

TABLE V
PROPOSED PRACTICES

Impacted agile practice	#	Proposed practice
Initial sprint planning	S22	Define obligations of individuals
Sprints	S22	Link processes to sprints
Release	S22	Progress report
Sprint planning meeting	S9	Brainstorming
	S14	Risk register
	S18	Identify the responsibilities of individuals
Daily meeting	S5	Risk Assessment Forum
	S6	Automatic agents
	S12	Impediment matrix
	S19	Automatic agents
	S20	Risk ranking
	S23	Risk list
Sprint review meeting	S6	Automatic agents
	S9	Brainstorming
	S19	Automatic agents
Sprint retrospective meeting	S21	Risk register
	S23	Risk list
Planning meeting	S23	Risk list
User stories	S6	Automatic agents
Continuous integration	S20	Risk ranking
Pair programming		
Face to face communication		
Flexible design		
Customer software demos		
Backlog management		
Iteration planning meeting	S7	Risk Poker
	S11	User stories repository
	S14	Risk register
Risk management meeting	S14	Qualitative risk analysis matrix
		Risk decomposition structure
		Risk cards
Kanban board	S8	Risk distribution
	S14	Risk closing
Work planning (Kanban)	S14	Risk register
		Qualitative risk analysis matrix
		Risk decomposition structure
		Risk cards
–	S1	Feedback loop
	S2	Identify responsibilities
	S3	Practice recommending tool
	S4	Initial meeting
	S10	AR Rank
	S13	Model-driven requirements
	S15	Risks checklist
		Brainstorming
		Analysis charts
		Forms
	S16	Quiz
	S17	Predictive risk identification

In S5, the Risk Assessment Forum (RAF) is proposed to be applied weekly in daily meetings. Thus, the development team and the Scrum Master can increase the identified risks and manage them. Study S9, inserted two brainstorming sessions, after the Sprint planning meeting to identify potential risks and in the Sprint review meeting to risks documentation. The practices proposed in S23 provide heuristics that facilitate risk analysis, prioritizing resolutions, and linking them into an overall plan. The proposed risk management process also involves team members in several informal knowledge-sharing exercises assisting decision-making and forming a risk list with their respective resolutions.

The Risk Analysis practice, proposed in S12, is defined for the XP method to reduce risks of user story overload by providing several alternative plans to improve negotiations between different stakeholders, promoting a deeper understanding and helping to choose a development plan with the greatest chance of being implemented on time.

In S8, authors propose an intervention in the Kanban workflow. In this new practice, identified risks are distributed to team members with defined roles. This provides a clear view of each person's tasks and responsibilities regarding risks. Using selected risk factors, study S17 developed models to predict whether a risk will cause a delay. If so, the model also determines the risk impact and the probability of occurrence.

Q4. What types of risks are managed?

The selected primary studies identify a total of 230 risks. Due to this large number of risks described in different ways, we decided to group them using a well known risks taxonomy [40], [41], [42] that provides three risk classes, its elements and attributes. We have collected all risks reported in the selected primary studies, classified according to the taxonomy, and summarized in Table VI. The complete list of risks, its sources and our chosen classifications is available at: bit.ly/36i7Wby.

The primary studies that reported the highest number of risks were [S2], [S20], and [S23]. Studies [S2] and [S20] were the works that have more risks classified in different attributes (25), followed by [S23], with risks classified in 19 attributes.

The attribute with the most risks occurrences was Schedule, with occurrences in 9 primary studies (39%), followed by other attributes from Budget and Staff (6 - 26%). The element with most risks occurrences was Requirements, with occurrences in 11 primary studies (48%). The class with the greatest number of occurrences was Development Environment, with occurrences in 13 primary studies (57%).

V. DISCUSSION

The results of this secondary study summarize information on the application of explicit risk management practices in agile software development methods.

As the wide majority of the selected studies were applied in industry with reported benefits, this raises evidence of the adequacy of explicit risk management practices in agile methods. All selected primary studies report positive impacts of introducing explicit risk management practices, with 10 studies (43%) [S1, S2, S3, S6, S12, S14, S15, S18, S19, S21] reporting positive impacts without compromising the "agility" of the agile methods.

The vast majority (61%) of the studies applied Scrum, confirming its global trend as the main agile method used

TABLE VI
CLASSIFICATION OF IDENTIFIED RISKS

Class	Element	Attribute	Studies
Product Engineering	Requirements	Stability	S2, S10, S17, S20, S21
		Completeness	S23
		Clarity	S2, S10, S20, S21, S23
		Validity	S16, S23
		Feasibility	S1
		Precedence	S2, S10, S17, S20, S23
		Scale	S6, S9, S14
	Design	Functionality	S13
		Performance	S13
		Testability	S2, S10, S20
		Hardware Constraints	S9
	Code and Unit Test	Feasibility	S21, S23
		Testing	S2, S10, S20
	Integration and Test	Environment	S1, S14
		Product	S2, S9, S10, S17, S20
		System	S2, S20
	Engineering Specialties	Maintainability	S6, S14
		Security	S9, S13, S21
		Specifications	S9, S16
Development Environment	Development Process	Suitability	S2, S10, S20
		Process Control	S2, S10, S12, S17, S20
		Familiarity	S15, S23
		Product control	S2, S14, S20
	Development System	Capacity	S2, S20
		Reliability	S9, S17
		Familiarity	S6, S9, S15, S23
		Deliverability	S17
	Management Process	Planning	S2, S10, S20, S21, S23
		Project Organization	S6, S16
		Management Experience	S23
		Program Interfaces	S2, S10, S14, S20, S23
	Management Methods	Personnel Management	S1, S2, S6, S23
		Quality Assurance	S12, S14
		Configuration Management	S20, S23
	Work Environment	Quality Attitude	S20
		Cooperation	S2, S6, S10, S17, S20
		Communication	S2, S6, S15, S20, S23
		Morale	S2, S6, S14, S23
Program Constraints	Resources	Schedule	S2, S6, S9, S10, S12, S16, S17, S20, S23
		Budget	S2, S10, S14, S16, S20, S23
		Staff	S2, S9, S14, S16, S20, S23
	Contract	Type of Contract	S2, S20
		Dependencies	S2, S14, S20, S23
	Program Interfaces	Customer	S2, S6, S14, S20, S23
		Corporate Management	S2, S10, S20
		Vendors	S2, S20
		Politics	S2, S14, S23

[5]. In contrast, only one study (4%) implemented DSDM, possibly indicating a tendency to disuse of this method.

The existent agile practices most affected by explicit risk management processes were the Daily Meeting and Iteration Planning Meeting, raising evidence that the incorporation of risk management practices especially affects the identification and monitoring of risks, corroborating other results reported in the literature [13].

In addition to the practices, we also have extracted and classified the managed risks. Most studies reported risks related to requirements and communication. The highlight the "delay", which affects 39% (9) of the selected studies.

A. Threats to validity

We have identified potential threats and applied mitigation strategies to minimize impacts on the outcomes following [43].

To reduce the risk of incomplete searches, we have selected, reviewed and tested search terms and also applied the Snowballing technique, which resulted in additional studies.

The number of studies, trend of publishing positive results, and empirical quality of most studies may affect the validity of the conclusions, as we decided to include studies with low empirical evidence to spot trends of topics being worked [12].

VI. CONCLUSION

Considering the lack of risk management processes in agile methods, this paper presents a Systematic Literature Mapping on how software organizations integrate explicit risk management practices into agile methods. We selected 23 primary studies following a defined research protocol.

The data collected indicate that the most used research method is case study. Selected primary studies are mostly applied in the industry using Scrum. The more frequently adapted agile practices are Daily Meeting and Iteration Planning Meeting with the introduction of specific risk management practices such as Risk Poker, Risk Ranking and Risk Register. The risks most frequently identified by studies are related to schedule and communication.

According to the results, explicit risk management practices provided benefits to the agile projects such as the increase in the number of identified risks and the choice of more effective corrective actions, improving team communication and the visibility of impediments, in addition to anticipating problems.

Therefore, the explicit inclusion of risk management practices can help the management of risks in agile projects without hurting the principles of agility, reducing its negative impact, and increasing the chances of success of the projects.

REFERENCES

[1] P. Bourque and R. E. Fairley, Eds., *SWEBOK: Guide to the Software Engineering Body of Knowledge*, version 3.0 ed. IEEE Computer Society, 2014. [Online]. Available: http://www.swebok.org/

[2] F. Hayat, A. U. Rehman, K. S. Arif, K. Wahab, and M. Abbas, "The influence of agile methodology (scrum) on software project management," in *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2019, pp. 145–149.

[3] A. Albadarneh, I. Albadarneh, and A. Qusef, "Risk management in agile software development: A comparative study," in *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2015, pp. 1–6.

[4] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 6th ed. Project Management Institute, 2017.

[5] Digital.ai, *15th annual state of agile report*. Digital.ai, 2021.

[6] M. Hammad, I. Inayat, and M. Zahid, "Risk management in agile software development: A survey," in *2019 International Conference on Frontiers of Information Technology (FIT)*, 2019, pp. 162–1624.

[7] L. Xiaosong, L. Shushi, C. Wenjun, and F. Songjiang, "The application of risk matrix to software project risk management," in *2009 International Forum on Information Technology and Applications*, vol. 2, 2009, pp. 480–483.

[8] M. Esteki, T. J. Gandomani, and H. K. Farsani, "A risk management framework for distributed scrum using prince2 methodology," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 3, pp. 1299–1310, 2020.

[9] E.-M. Schön, D. Radtke, and C. Jordan, "Improving risk management in a scaled agile environment," in *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer International Publishing, 2020, pp. 132–141.

[10] F. Hayat, M. W. Anwar, F. Azam, and A. Kiran, "A sysml-based approach for requirements risk management and change control," in *Proceedings of the 2019 11th International Conference on Information Management and Engineering*, ser. ICIME 2019, 2019, p. 20–24.

[11] A. Sousa, J. P. Faria, and J. Mendes-Moreira, "An analysis of the state of the art of machine learning for risk assessment in software projects," in *Proceedings of the 33rd International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2021, pp. 1–10.

[12] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[13] M. Vieira, J. C. R. Hauck, and S. Matalonga, "How explicit risk management is being integrated into agile methods: Results from a systematic literature mapping," in *19th Brazilian Symposium on Software Quality*, ser. SBQS'20, 2020.

[14] S. Y. Chadli and A. Idri, "Identifying and mitigating risks of software project management in global software development," in *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, 2017, p. 12–22.

[15] Z. Podari, A. F. Arbain, N. Ibrahim, D. N. Abang Jawawi, W. M. Nasir Wan Kadir, and A. M. Fahmi, "Systematic literature review on global software development risks in agile methodology," in *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, 2020, pp. 231–236.

[16] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE'08. Swindon, GBR: BCS Learning amp; Development Ltd., 2008, p. 68–77.

[17] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14, 2014.

[18] S. M. Al-Saleem and H. Ullah, "A comparative analysis and evaluation of different agile software development methodologies," in *International Journal of Computer Science and Network Security*, vol. 15, no. 7, 2015, pp. 39–45.

[19] M. Turner, "Digital libraries and search engines for software engineering research: An overview," *Keele University, UK*, 2010.

[20] P. Crowe, A. Mostashari, M. Mansouri, and R. Cloutier, "9.2.1 reference framework and model for integration of risk management in agile systems engineering lifecycle of the defense acquisition management framework," *INCOSE International Symposium*, vol. 19, no. 1, pp. 1391–1405, 2009.

[21] B. G. Tavares, M. Keil, C. E. Sanches da Silva, and A. D. de Souza, "A risk management tool for agile software development," *Journal of Computer Information Systems*, vol. 61, no. 6, pp. 561–570, 2020.

[22] J. M. P. Carvallo, H. Oktaba, and E. R. Hernández, "Risk assessment forum," in *2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2018, pp. 160–164.

[23] E. Odzaly, D. Greer, and D. Stewart, "Agile risk management using software agents," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, p. 823–841, 2018.

[24] S. Ghazali, S. Salirti, I. Inayat, and S. h. Ab hamid, "A risk poker based testing model for scrum," *Computer Systems Science and Engineering*, vol. 33, pp. 169–185, 05 2018.

[25] V. Dorca, R. Munteanu, S. Popescu, A. Chioreanu, and C. Peleskei, "Agile approach with kanban in information security risk management," in *2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2016, pp. 1–6.

[26] M. Hammad and I. Inayat, "Integrating risk management in scrum framework," in *2018 International Conference on Frontiers of Information Technology (FIT)*, 2018, pp. 158–163.

[27] R. Agrawal, D. Singh, and A. Sharma, "Prioritizing and optimizing risk factors in agile software development," in *2016 Ninth International Conference on Contemporary Computing (IC3)*, 2016, pp. 1–7.

[28] X. Dong, Q.-S. Yang, Q. Wang, J. Zhai, and G. Ruhe, "Value-risk trade-off analysis for iteration planning in extreme programming," in *2011 18th Asia-Pacific Software Engineering Conference*, 2011, pp. 397–404.

[29] L. Ribeiro, C. Gusmao, W. Feijo, and V. Bezerra, "A case study for the implementation of an agile risk management process in multiple projects environments," in *PICMET '09 - 2009 Portland International Conference on Management of Engineering Technology*, 2009, pp. 1396–1404.

[30] L. G. Cuong, P. D. Hung, N. L. Bach, and T. D. Tung, "Risk management for agile projects in offshore vietnam," in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, ser. SoICT 2019, 2019, p. 377–384.

[31] B. Freimut, S. Hartkopf, P. Kaiser, J. Kontio, and W. Kobitzsch, "An industrial case study of implementing software risk management," in *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2001, p. 277–287.

[32] O. Mizuno, T. Kikuno, Y. Takagi, and K. Sakamoto, "Characterization of risky projects based on project managers' evaluation," in *Proceedings of the 22nd International Conference on Software Engineering*, ser. ICSE '00, 2000, p. 387–395.

[33] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Characterization and prediction of issue-related risks in software projects," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 280–291.

[34] J. Nyfjord and M. Kajko-Mattsson, "Outlining a model integrating risk management and agile software development," in *2008 34th Euromicro Conference Software Engineering and Advanced Applications*, 2008, pp. 476–483.

[35] E. Odzaly, D. Greer, and D. Stewart, "Lightweight risk management in agile projects," in *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2014, pp. 576–581.

[36] S. V. Shrivastava and U. Rathod, "A risk management framework for distributed agile projects," *Information and Software Technology*, vol. 85, pp. 1–15, 2017.

[37] E. Alharbi and M. R. Qureshi, "Implementation of risk management with scrum to achieve cmmi requirements," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 6, pp. 20–25, 09 2014.

[38] M. Mousaei and T. J. Gandomani, "A new project risk management model based on scrum framework and prince2 methodology," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 4, 01 2018.

[39] S. Ghobadi and L. Mathiassen, "Risks to effective knowledge sharing in agile software teams: A model for assessing and mitigating risks," *Information Systems Journal*, vol. 27, no. 6, pp. 699–731, 2017.

[40] M. Carr, S. Konda, I. Monarch, C. Walker, and F. Ulrich, "Taxonomy-based risk identification," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., 1993.

[41] S. Sundararajan, M. Bhasi, and P. Vijayaraghavan, "Variation of risk profile across software life cycle in is outsourcing," *Software Quality Journal*, vol. 27, p. 1563–1582, 12 2019.

[42] H. A. Abdulbaqi, A. S. A. Jabar, and Z. S. A. Jabar, "Integrated software project risks method based on pdf-ann techniques," *International Journal of Civil Engineering and Technology (IJCIET)*, p. 1094–1102, 2018.

[43] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, "A map of threats to validity of systematic literature reviews in software engineering," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, 2016, pp. 153–160.

LSTMcon: A Novel System of Portfolio Management Based on Feedback LSTM with Confidence

Xinjia Xie[1,⋆], Shun Gai[1,⋆], Yunxiao Guo[2,⋆], Boyang Wang[1,⋆], Han Long[2,†]
[1] College of Computer Science, National University of Defence Technology, Changsha, China
[2] College of Liberal Arts and Sciences, National University of Defence Technology, Changsha, China
Email Address: {xinjiaxie@yeah.net; shun12581@nudt.edu.cn; guoyunxiao.nudt@hotmail.com; wangboyang16@nudt.edu.cn; longhan@nudt.edu.cn}

***Abstract*—Trading carries a substantial amount of risk and making adequately informed decisions cannot be overemphasized. In order to propose a more reasonable strategy on portfolio arrangement, we design LSTMcon, a two-stage system that consists of a assets price prediction model and a decision-making strategy based on ensemble rules. As for next-day price prediction, we implement an LSTM model with feedback mechanism and devise a series of training settings. The feedback mechanism uses the deviation between predicted price and actual price to correct the prediction result from LSTM. To decrease the transaction cost, we design a three-day trading period and adopt an iterative prediction approach. Our model achieves the accuracy of 98.5% on GOLD and 98.8% on BTC finally. In addition, we devise a decision-making system after getting the predicted data. We modify the predicted price by giving everyone a certain confidence level based on three approaches (reward and punishment mechanism, sequential days rules, historical price relying). We combine these rules and give a comprehensive confidence level to weigh the predicted price. Subsequently, we summarize the transactions into 8 trading operations, input the modified price and automatically compare the hypothetical return of these eight operations. Then, output the operation with largest return as today's decision. We compare the returns and transaction costs of comparative systems, and demonstrate our strategy with effectiveness.**

***Index Terms*—Price Prediction, Decision-making, LSTM, Portfolio Management, Knowledge Engineering**

I. Introduction

Market traders buy and sell volatile assets frequently, with a goal to maximize their total return. Nowadays, it is not difficult for us to find trading strategies suitable for our preferences in many academic articles or forums [1]. However, it is still a problem of how to distinguish the good and bad of these strategies and avoid making some common mistakes, such as survivorship bias, look-ahead bias, and trading cost [2].

With the popularity of Internet resource search, quantitative trading emerges as the times require [3]. It refers to using advanced mathematical models to replace human subjective judgment, and selecting a variety of high probability events that bring excess returns from huge historical data to formulate strategies. For example, in some simple quantitative analysis [4], some statistical tests are carried out on the close price of financial instruments. More complex systems consider more information to improve the accuracy of price prediction [5].

In this paper, we deliver a novel system of portfolio management based on a two-stage model consisting of a financial assets price prediction model and decision-making strategy based on ensemble rules (Fig. 1). We apply a LSTM with Feedback Mechanism for prediction and set aside a period of half a year used to train a preliminary LSTM which constantly carries out training with real assets price of the days we have predicted and modifies its parameters. It can be generalized to other asset datasets in addition to gold and bitcoin. As for decision-making, we introduce several rules to modify the predicted assets price by giving everyone a certain confidence level in order to propose a more reasonable strategy. The remainder of this paper is organized as follows: After concluding related work in Sec. II, Sec. III discusses details of our portfolio management system. Experiment settings and results analysis are revealed in Sec. IV and we summarize this paper with future work discussed finally.

II. Related Work

There are some traditional time series models applied to portfolio management based on typical methods in mathematics. These methods include multivariate analysis [6], dependence learning [7] and transitional models [8]. In spite of their widely use in scientific experiments, they were proposed early and incapable of modeling the aforementioned huge data and complex features.

Recently, researchers apply more deep learning models for time series prediction tasks, such as recurrent neural networks (RNN) based method with a dual-stage attention [9] to select relevant driving series, and convolution neural network (CNN) based method like TCN [10] to capture effective historical data of price fluctuation. They are regarded as good trials on assets price prediction. Related work also pays attention to improve the accuracy by utilizing external information which may influence the market and effect assets price [5]. Clearly, a deep learning model does show a better performance on big data and take more factors into consideration.

⋆ Equal Contribution
† Corresponding Author
DOI reference number: 10.18293/SEKE2022-160

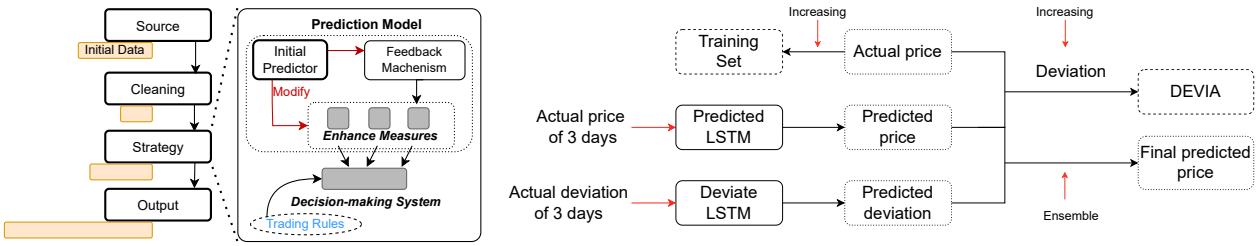


Fig. 1: The framework of our system: (a) Quantitative trading process we simulate in this paper: data flows in the prediction model and decision-making strategy; (b) The detail process of our financial assets price prediction model: LSTM with Feedback Mechanism and utilised datasets.

We conclude with these SOTA models from both computer scientists and economists in real-world investment, but they are difficult to be directly formed as a strategy. Among them, RNN[11], [12], [13] is one of the most powerful models. Compared with other neural network, results of each layer in RNN are not independent, but relate to results of the previous layer and current input. However, computational complexity would increase exponentially, resulting in a significant increase of model training consumption. Due to the assumption that cash from selling gold or bitcoin could be used to buy assets on the same day, there is no necessity to consider multi-step prediction. In this paper, we start with the variant of RNN: Long Short-Term Memory (LSTM) [14] and some measures are taken on the single LSTM for improvements.

III. Method

A. Financial Assets Price Prediction System

1) LSTM with feedback mechanism: LSTM is specially designed to solve the long-term dependence problem of general RNN, thus it helps with prediction of sequential data, of which a good example is constituted by daily price of financial assets.

We set the training rules as follows:

- Considering the transaction cost, we design a three-day trading period, that the trader buys or sells assets using data of every three days. Therefore, we choose the iterative prediction approach. Our model predicts the daily rise or fall of three days at a time and we make one trading decision based on these data (the reason for choosing 3 is in Sec. IV-E with experiments).
- Due to prediction based on data up to that day, we cannot use the later data to train our model. We set aside a stable period of half a year. For example, we use data from September 2016 to February 2017 to preliminarily train an LSTM but the training set size increases day by day.
- The maximum return may be closely related to the exact predicted price. Therefore, we device Feedback Mechanism to modify our model. After comparing the predicted value with the real value, we return the deviation to LSTM by Feedback Mechanism, and modify parameters.

The final structure of our prediction model is shown in Fig. 1 (b). Feedback Mechanism is actually a deviate LSTM. Based on the predicted LSTM, we add it to return the deviation between predicted price and actual price. First, we train a preliminary LSTM for predicting future prices from six-month data for iterative prediction. Its training set is increasing day by day, and it constantly carries out training, that is, modifying parameters to improve the prediction performance. Next, we accumulate the deviation between the predicted price and the actual price to form a training set named DEVIA. DEVIA is used to train the deviate LSTM to input the deviation of several days and obtain the deviation of the next day, which is used to correct the prediction result of the predicted LSTM. The outputs of the two models are added together, i.e. predicted price + predicted deviation, as our final predicted value.

B. Decision-making Strategy Based on Ensemble Rules

1) Predicted Price Modification: In order to reduce potential risks of prediction models, we consider how much confidence is given to predicted results. To evaluate the confidence, we introduce several approaches which help filter opportunities with low winning rate, which is very difficult for machines.

Since we have adjusted the predictor by Feedback, the results are actually reliable and we set the lowest confidence as 0.94 (the reason for choosing 0.94 is in Sec. IV-E with experiments). with the maximum confidence of 1. Each evaluation method gives a confidence to predicted results. We combine these confidences to give a comprehensive one to weigh the results for next decision-making. Confidence of different days would continuously change in [0.94, 1].

(a) Reward and Punishment

We study the process of real neural networks generating strategies through neural circuits. After animals successfully hunts with a certain method, they would incline to use this method in the next hunt, and abandon the failed one. It mainly depends on the reward and punishment mechanism.

We manage to set **Reward and Punishment** in our model, which rewards and punishes the confidence of predicted price. At the beginning, confidence is relatively low. As the number of correct predictions increases, it would rise linearly; otherwise, reduce it appropriately.

(b) Transaction rules

TABLE I: Model Comparsion

Technique	RP Mechanism	Sequential Days	Historical Price
Range	[0,2000]	[1000,2000]	[0,2000]
Initial Value	0	2000	2000
Change per day	±100	±100	±100

From the perspective of economics, we directly incorporate some summarized laws into our system, such as the two famous economic principles in the follows [15]:

- People make decisions after comparing costs and benefits. When costs and benefits change, people's decisions also change. So they respond to incentives, which can be either artificial or the result of natural change.
- The market fluctuates and the fluctuation has a direction. Although there are differences between strong and weak fluctuations, fluctuations is cyclical in a long term.

As for the first principle, we believe that the trader's mentality of buying, holding, or selling his assets is in a dynamic process of days, even if the change ratio is the same. When the price rises (falls) for several **Consecutive Days**, traders are more likely to sell (buy) than on the first day, which is a right way to invest. We suppose the predicted price is rising continuously for several days. On the first day, we give almost full confidence, because it is a normal fluctuation. Confidence would decrease with days passing, since it is difficult to imagine that an asset will rise for ten days or more.

For next principle, we adjust confidence according to **Historical Price** since the fluctuation is cyclical. If the price falls to historical bottom (all-time low), we tend to think that it would rise. Stay alert to consider whether we sell the assets or hold on when the predicted price is close to historical peak.

(c) The combination

We write a scoring program with three indicators that we proposed: **Reward and Punishment (RP)**, **Conservative Days** and **Historical Price**. The rules are shown in Table I and these parameters are the best determined by repeated tests in subsequent experiments.

As for **RP**, the initial value is 0 and it increases by 100 every time the prediction is correct, and decrease when false. For **Consecutive Days**, experimental results give it a large minimum value, which means we rely on it very much. On the first day of rising, it is 2000 but decreases by 100. For **Historical Data**, when it is between historical maximum and minimum, we give it 2000. Every time it falls outside the range, 100 points are deducted.

The total score is within [1000, 6000]. We map them to [0.94, 1] to obtain a confidence in [0.94, 1]. Let RP be the score given by **RP**, Seq be the score of **Consecutive Days** and His of **Historical Data**. The mapping relationship is:

$$Conf = (His + Seq + RP - 1000)/50000 + 0.94 \quad (1)$$

Experiments show that the composition of the confidence is very reasonable. At the initial stage of prediction, the proportion of scores given by constructive days is the highest because of the lack of samples, on which RP mechanism and historical data bases. In the medium term, the proportion of RP mechanism rises quickly. When predicting the final data, the score of confidence is almost determined by historical price.

2) Decision Tree: We summarize the transactions in each day into 8 trading operations as follows:

- $B \rightarrow G$, sell bitcoin and buy gold
- $G \rightarrow B$, sell gold and buy bitcoin
- $B\downarrow$, $G\downarrow$, all sold
- B— , G— , no operations
- $B\uparrow$, G— , buy bitcoin, no operations on gold
- B— , $G\uparrow$, no operations on bitcoin, buy gold
- $B\downarrow$, G— , sell bitcoin, no operations on gold
- B— , $G\downarrow$, no operations on bitcoin, sell gold

We simulate these eight trading operations, as shown in the following formulas. On the left of the equation is the hypothetical return. In each case, the whole return from the operation is subtracted from the cost of the corresponding operation (transaction cost and income that can be obtained if the operation is not carried out), and then the hypothetical return are obtained. After getting the predicted price of the next day, our algorithm automatically compares the hypothetical return of these eight operations. Then, output the operation with largest return as today's decision.

- $B_1 = (P_{G_{d+3}} - R_{G_d}) * (T_{B_d} - 0.02 * T_{B_d} - 0.01 * 0.98 * T_{B_d})/R_{G_d} - (0.02 * T_{B_d} + 0.01 * 0.98 * T_{B_d} + (P_{B_{d+3}} - R_{B_d}) * H_B) + (P_{G_{d+3}} - R_{G_d}) * H_G$
- $B_2 = (P_{B_{d+3}} - R_{B_d}) * (T_{G_d} - 0.01 * T_{G_d} - 0.02 * 0.99 * T_{G_d})/R_{B_d} - (0.01 * T_{G_d} + 0.02 * 0.99 * T_{G_d} + (P_{G_{d+3}} - R_{G_d}) * H_G) + (P_{B_{d+3}} - R_{B_d}) * H_B$
- $B_3 = -1 * (T_{G_d} * 0.01 + T_{B_d} * 0.02)$
- $B_4 = (P_{B_d} - R_{B_d}) * H_B + (P_{G_d} - R_{B_d}) * H_G$
- $B_5 = (P_{B_{d+3}} - R_{B_d}) * 0.98 * H_D/R_{B_d} + (P_{B_{d+3}} - R_{B_d}) * H_B + (P_{G_{d+3}} - R_{G_d}) * H_G$
- $B_6 = (P_{G_{d+3}} - R_{G_d}) * 0.99 * H_D/R_{G_d} + (P_{B_{d+3}} - R_{B_d}) * H_B + (P_{G_{d+3}} - R_{G_d}) * H_G$

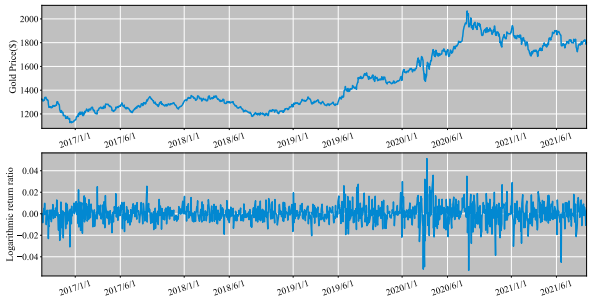


Fig. 2: A example: the price fluctuation curve of Gold and the corresponding Logarithmic return ratio of each day.

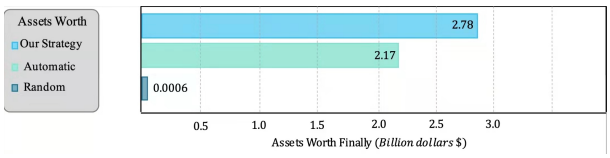
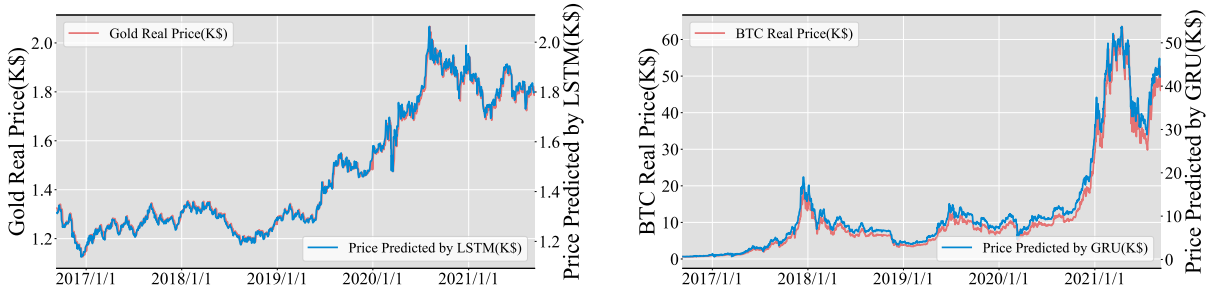
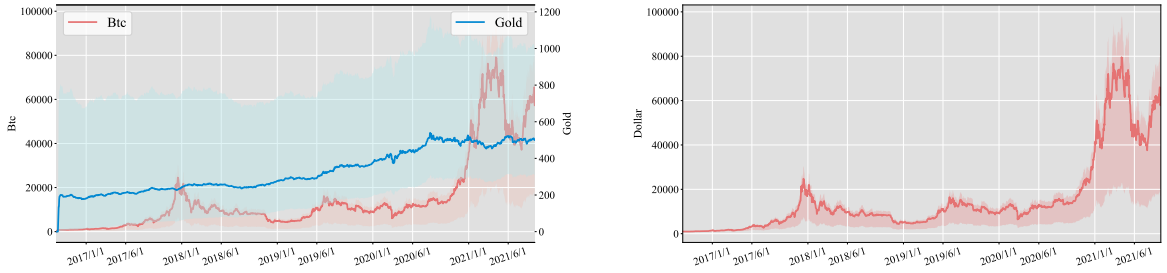


Fig. 3: Assets Worth Finally on three systems.



(a) (b)

Fig. 4: Predicted price fluctuation curves of (a) LSTM with feedback on GOLD, (b) LSTM with feedback on BTC.



(a) (b)

Fig. 5: In the random system, (a) amount of gold and bitcoin we hold in a timeline; (b) assets worth in a timeline

- $B_7 = (P_{G_{d+3}} - R_{G_d}) * H_G - 0.02 * T_{B_d}$
- $B_8 = (P_{B_{d+3}} - R_{B_d}) * H_B - 0.01 * T_{G_d}$

Take $Branch_8$ as an example which represents no operations on bitcoin and selling gold. The final predicted return (assuming no operations on holding assets and all surplus cash to buy gold in the next three days) is equal to the three-day return on buying gold, and holding gold and bitcoin.

Suppose the situation: the assets price falls next day with the decline just greater than transaction cost, but it rises immediately the day after. In these cases, the decision from models directly using predicted price causes repeated jump and consumption, and our model effectively avoids the problem.

C. Comparative Systems

Although we already devise a decision-making strategy, we think it necessary to design two other simple decision-making systems for comparison. After leveraging existing predictor, we get a set of price fluctuation prediction results. We put these results into these systems and calculate the income.

The first is **Random System** that we devise based on our prediction model. We design a random algorithm as follows: after we gain the prediction results, we input them into the algorithm and get a random value, the asset value we decide to buy or sell. We aim to make money and would not trade against price fluctuation, but are eager to get a return under random circumstances. In the experiment below, we run the program 1000 times to obtain the random mean return.

Next comes a simple **Automatic System**. This system is hardly added with any decision-making strategy. As long as the predicted decline next day exceeds transaction cost, we sell the assets, and vice versa. Unless both assets fall, we do not keep cash in wallets. If both assets rise or other complex conditions, we would allocate the percentage of each asset according to rules in Sec. III-B2.

IV. Experiments

A. Experimental Setup

1) Datasets: In this paper, we only take two assets: gold and bitcoin as examples. To predict the price based on data up to that day, we collected the price of gold and bitcoin during a five-year trading period from September 11, 2016 to September 10, 2021, and name them as GOLD and BTC. We supplement the default data of GOLD to improve the accuracy of subsequent prediction, and calculate the specific fluctuation ratio of assets price each day.

To gain a more intuitive understanding of price changes, we create a column that indicates the Logarithmic return ratio of each day. The price fluctuation follows a log-normal distribution with a more stationary characteristic [16].

2) Index: Based on the two comparative systems mentioned in Sec. III-C, we mainly use two indexes as follows.

Return: Assume that we have $1000 in the beginning as the initial capital, and we calculate how much our assets is worth after five-year prediction and decision-making.

Accuracy: Since the exact changing range is difficult to predict, the accuracy in this paper refers to whether we correctly predict it rises or falls.

TABLE II: Model Modify

Models	Accuracy_GOLD (%)	$Training\ MSE_{GOLD}$	$Test\ MSE_{GOLD}$
LSTM	98.2	1373.836482	1378.41921
LSTM with Feed Back	98.5	1321.472946	1299.382565
Models	Accuracy_BTC (%)	$TrainingMSE_{BTC}$	$TestMSE_{BTC}$
LSTM	98.5	7132.357321	31232.64239
LSTM with Feed Back	98.8	7037.683933	29475.83646

B. Assets Price Prediction

We assume that the real data has a certain law, and believe that LSTM learns some laws from the training set, but previous experiments indicate that there are some deviations between the laws contained in real data and learned by LSTM. Accordingly, we speculate that the gap between prediction results and real data also forms a certain law. Therefore, we add Feedback Mechanism (deviate LSTM in Sec. III-A) for predicting the gap to correct the predicted price.

As shown in Fig. 4, LSTM with Feedback simulates the whole price fluctuation almost perfectly even on BTC with exaggerated fluctuations. Table II better shows the performance of LSTM with Feedback and it behaves very improvement on all metrics compared to a single LSTM. Our model achieves the accuracy of 98.5% on GOLD and 98.8% on BTC, which demonstrate Feedback mechanism with highly effectiveness.

C. Three Decision-making Systems and Who is the Best

In general, we calculate how much our assets is worth finally. According to the prediction, **Random System** achieves \$0.0006 billion in five years, which is too little to show on the Fig. 3. **Automatic System** which makes decisions only based on the relationship between prediction and transaction cost shows a good performance of \$2.17 billion.

However, the market is difficult to predict and untenable to directly rely on computer models. Therefore, we introduce mature market rules in our decision-making model to modify the predicted price, and finally get \$2.78 billion.

1) The random system: The performance of **Random System** is worth mentioning (Fig. 5 in a timeline). After experimental data of 1000 random trials are averaged, the solid line is obtained. The shaded part indicates the fluctuation range of 1000 randomized trials. The assets value almost completely follows bitcoin. We guess the reason is the price of bitcoin fluctuates more suddenly and steeply than that of gold. Bitcoin has risen greatly in recent years, so the results are better. However, there are also great disadvantages. When bitcoin plummeted, our total assets decreased sharply. It shows that blind investment is problematic even with a correct prediction.

2) Our strategy: Fig. 6 shows the performance of our strategy with assets worth and different assets' ratio in a timeline. Since the effect of the other two models is relatively weak, we mainly compare our model with **Automatic System**.

The final assets value of our strategy is 28.1% higher than that of Automatic System, which fully shows the effectiveness of the laws we designed from different disciplines. As is shown in Fig. 6, the overall worth is rising almost consistently and the ratio of different assets is relatively uniform. Each color lasts a certain distance that indicates the ratio is relatively stable. There is no problem of frequent buying and selling of Automatic System, which leads to a large amount of handling charge.

TABLE III: The transaction cost and ratio of two models.

Models	Transaction cost	Return	Ratio (%)
Decision-making system	1.68	2.78	60.43
Automatic system	1.78	2.17	82.04

In addition, with the addition of well-designed decision-making methods, we can well avoid the characteristic that the value follows bitcoin in the random system. We discover that the value of assets has been rising over time without any decline, which shows our strategy the most stable one. Unlike the random system that always invests in bitcoin, when the predicted price of bitcoin falls, our strategy will buy a lot of gold to ensure that assets value are not lost.

D. Influence of Transaction Cost

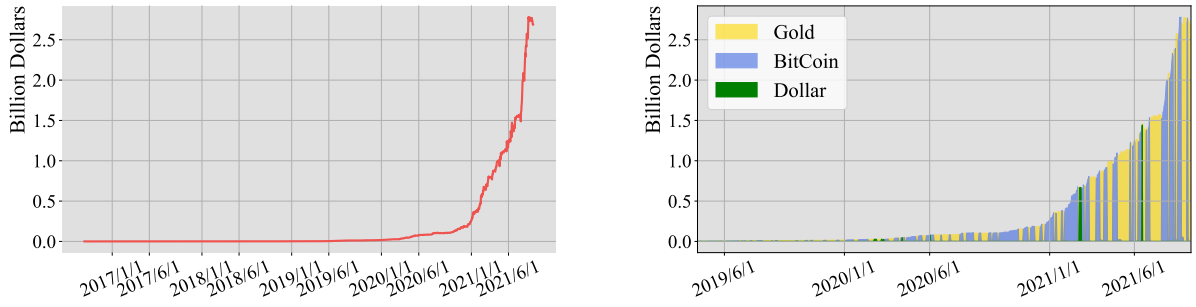
In this part, we compare our model mainly with **Automatic System**. We get the transaction cost through programming, and calculate the ratio between it and the actual return of the model. Table III below includes these information.

In this table, Automated System spends more on transaction fees during the investment period, meaning more daring actions. After introducting market rules, we spent less fees and achieved 27% higher profits than the automatic system. We conclude that machines tend to short-term benefits, while decision-making systems pay more attention to long-term benefits, which is also what economic principles tell us. It indicates that decision-making is significantly effective in portfolio management.

E. Parameters chosen

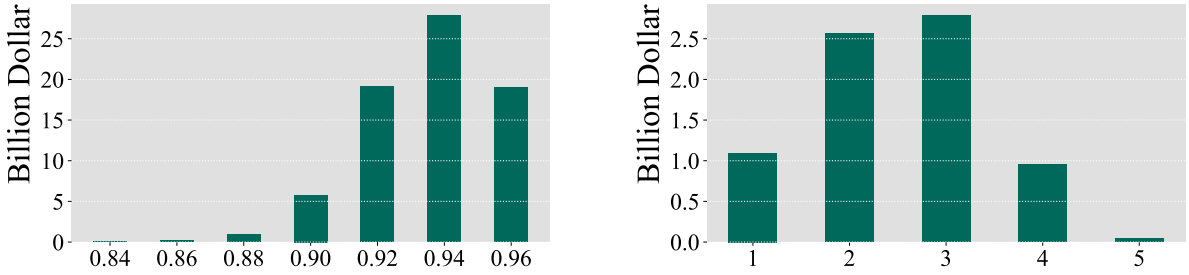
Fig. 7 (a) shows how much the assets worth finally under different lower confidence limits. In Sec. III-B, based on the predicted results of LSTM with a feedback mechanism, we set a confidence level for the predicted price by combining some economic principles and other methods. Confidence levels would inevitably effect the final results, so we need to find the most appropriate value. Experimental results show that when the confidence level is 0.94, the final assets price maximized.

For portfolio management, we make decisions with reference to predicted prices. In reality, there are often frequent changes in prices in a short term, which leads to unnecessary trading operations and increases the transaction cost. The problem can be effectively avoided by predicting prices of several days in the future, but it leads to inaccurate prediction when we predict prices for too many days. We conduct repeated experiments and finally discover in Fig. 7 (b) that predicting the price of next three days brings the best results.



(a) (b)

Fig. 6: In our strategy, (a) red line indicates how much is our assets worth in a timeline of five years; (b) ratio of different assets in our portfolio management during the same time.



(a) (b)

Fig. 7: Our assets worth under different parameters: (a) different lower confidence limits range in [0.84, 0.96]; (b) different number of days of which we predict assets prices and train iteratively.

V. Conclusion

For better portfolio arrangement, we established a novel system which contains a price prediction model and a decision-making model. We applied an LSTM model with Feedback Mechanism and achieved an excellent accuracy on two datasets. As for decision-making, we devised an ensemble method based on three approaches (reward and punishment mechanism, sequential days rules, historical price relying). to modify the predicted price by giving a confidence level. We summarized transactions into 8 trading operations and designed an algorithm to automatically output the operation with largest return. We calculate the returns and transaction costs of comparative systems, to demonstrate that our strategy is more reasonable. For future work, we can further combine external information, such as policies to enhance the system.

References

[1] Y. Fang, K. Ren, W. Liu, D. Zhou, and T. Y. Liu, "Universal trading for order execution with oracle policy distillation," 2021.

[2] A. A. Obizhaeva and J. Wang, "Optimal trading strategy and supply/demand dynamics," *Journal of Financial Markets*, vol. 16, no. 1, pp. 1–32, 2013.

[3] E. P. Chan, *Quantitative trading: how to build your own algorithmic trading business*. John Wiley & Sons, 2021.

[4] S. Bai, J. Z. Kolter, and V. Koltun, "Trellis networks for sequence modeling," 2018.

[5] R. Wang, H. Wei, B. An, Z. Feng, and J. Yao, "Commission fee is not enough: A hierarchical reinforced framework for portfolio management," 2020.

[6] M. H. Chen, J. G. Ibrahim, and Q. M. Shao, "Maximum likelihood inference for the cox regression model with applications to missing covariates," *Journal of Multivariate Analysis*, vol. 100, no. 9, pp. 2018–2030, 2009.

[7] J. Durbin, S. J. Koopman, and O. U. P. (OUP), *Time Series Analysis by State Space Methods*. Time series analysis by state space methods /, 2012.

[8] A. Beber and M. W. Brandt, "Resolving macroeconomic uncertainty in stock and bond markets," *NBER Working Papers*, vol. 13, no. 1, pp. 1–45, 2006.

[9] Song, H. Chen, G. Jiang, and Y. Qin, "Dual stage attention based recurrent neural network for time series prediction," 2018.

[10] H. Wang, H. Ahluwalia, R. A. Aliaga-Diaz, and J. H. Davis, "The best of both worlds: Forecasting us equity market returns using a hybrid machine learning – time series approach," *Social Science Electronic Publishing*.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[12] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[13] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," *arXiv preprint arXiv:1412.6632*, 2014.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] N. G. Mankiw, *Principles of economics*. Cengage Learning, 2014.

[16] R. S. Hudson and A. Gregoriou, "Calculating and comparing security returns is harder than you think: A comparison between logarithmic and simple returns," *International Review of Financial Analysis*, vol. 38, pp. 151–162, 2015.

Research on Identification and Refactoring Approach of Event-driven Architecture Based on Ontology

Li WANG[1,2]
[1]School of Computer Science and Engineering
Southeast University
Nanjing, China
[2]Jiangsu Automation Research Institute
Lianyungang, China
wangli1218@seu.edu.cn

Xiang-long KONG
School of Computer Science and Engineering
Southeast University
Nanjing , China
xlkong@seu.edu.cn

Xiao-fei WANG
NARI Group corporation
Nanjing, China
wangxiaofei@sgepri.sgcc.com.cn

Bi-xin LI ‡
School of Computer Science and Engineering
Southeast University
Nanjing , China
bx.li@seu.edu.cn

***Abstract*—Event-driven architecture is one of the common software architecture patterns. In the process of software evolution, the deviation and corrosion often occur to architecture, which leads to larger deviation between actual software architecture and design architecture. Therefore, it is of great significance to study the approach of software architecture identification and refactoring. To solve this problem, we propose an identification and refactoring approach of event-driven based on ontology, i.e., IRABO. We evaluated IRABO on 50 open-source projects and the results show that it performs effectively and efficiently.**

Keywords-Event-driven Architecture; Architecture Identification; Architecture Refactoring

I. INTRODUCTION

Appropriate pattern can solve the design problem of software architecture[1]. Event-driven architecture is a very popular architecture pattern at present, which is usually used in systems that require high agility and quickly response[2]. The event-driven architecture can fulfill that requirement quite well. But, in the process of software evolution, many factors may make the software architecture deviate from the original design, such as the change of requirements, the improvement of functions, et al. So, it is necessary to refactor the architecture. Architecture patterns provide a good direction for refactoring [3].

In this paper, we propose identification and refactoring approach of event-driven architecture based on ontology, i.e., IRABO, which consists of two parts. We firstly extract the dependency information from source code to build the program dependency graph. Then we convert the program dependency graph into RDF (The Resource Description Framework) triples to build the ontology of instance layer. Finally, we use the event-driven architecture usage specification to locate the refactoring point in the identification result, and refactor the software architecture.

To evaluate the effectiveness accuracy and efficiency of IRABO, we conduct experiments on 50 open-sourced projects with manual analysis approach. The results show that IRABO performs much better in terms of accuracy and effectiveness efficiency in our experiments. In summary, our paper makes the following novel contributions:

We put forward the ontology-based event-driven architecture pattern identification approach and the architecture refactoring approach based on event-driven architecture.

We build the experiment for ontology-based event-driven architecture identification and refactoring to verify the ontology-based event-driven pattern identification and refactoring approach.

II. APPROACH

In this section, we present the details of the identification and refactoring approach of event-driven based on ontology, i.e., IRABO. The technique comprises two main steps, identification approach of event-driven architecture based on ontology, refactoring approach of event-driven architecture based on ontology.

A. Identifying Event-driven Architecture Based on Ontology

Event-driven architecture identification based on ontology is essentially a process of abstract matching between source code and event-driven architecture. As presented in Fig.1, First, we use the source code analysis tool to extract the dependency information. Second, we use ontology to descript the dependency information to construct instance layer ontology; meanwhile, we use ontology to describe the structural behavior characteristics of event-driven architecture to construct concept layer ontology. The instance layer ontology and concept layer ontology form the Ontology Knowledgebase. We use ontology inference engine to process the Ontology knowledgebase to obtain the instance of event-driven architecture. Compared with other semi-automatic or manual approaches, IRABO can improve the accuracy and automation of event-driven architecture identification.

1) Construction instance layer ontology:

In this paper, the object of identification is Java projects. We choose JDT to enerate an Abstract Syntax Tree, i.e., AST.

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61872078)
DOI:10.18293/SEKE2022-013

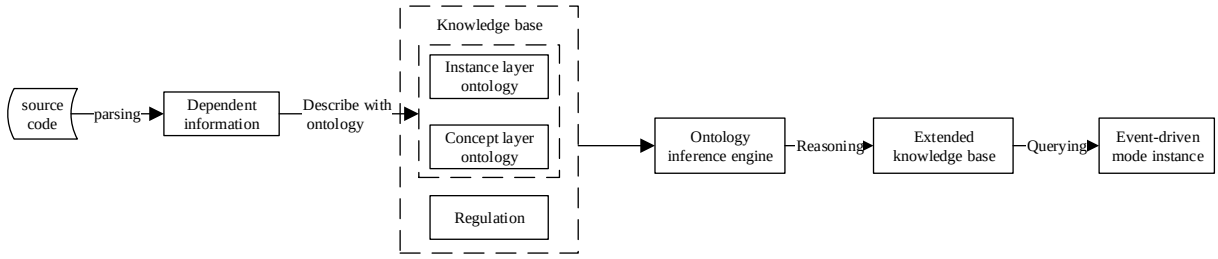


Figure 1. Event-driven architecture identification based on ontology

We extract the dependency information by traversing the ADT to build a dependency graph. As shown in Fig 2, the node represents the program entities, and the directed edge represents the dependency between program entities. We convert the nodes and directed edges into RDF triples set.

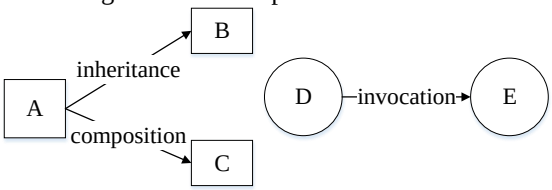


Figure 2. RDF triple set

2) Construction concept layer ontology:

In this paper, we choose the common ontology building Jena to build ontology of the event-driven architecture. First, we use ontology to describe the observer pattern and its specific application in event-driven architecture, thus indirectly describing the behavior characteristics of event-driven architecture. Second, we use ontology to describe the component reuse behavior of the event-driven system to the event-driven framework. riven architecture usage specification to locate the refactoring point in the identification result, and refactor the software architecture.

The event-driven architecture has three components: event, listener and event source. The listener acts as the observer, and the event source acts as the observed[3][4] [5]. The behavior characteristics of the event-driven architecture are shown in Fig 3.

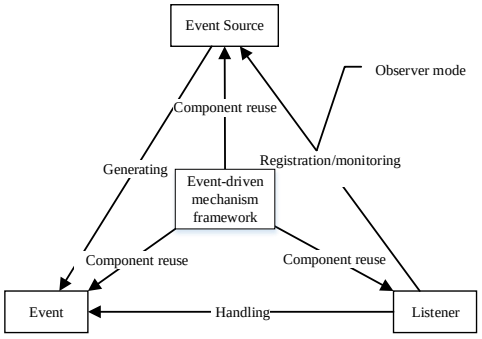


Figure 3. Behavior characteristics of event-driven architecture

An important behavioral feature of event-driven architecture is the component reuse behavior of event-driven framework, as shown in Fig 4. When programmers develop event-driven systems under the framework of event-driven mechanism, they only need to define the listeners through the listener interface, and inherit the sensible operating components under the framework to define their own event sources and the event classes under the framework to define their own events[6]. We describe the event-driven architecture indirectly by describing observer pattern and event-driven framework. We build ontology to describe the component reuse behavior of the event-driven framework on the ontology building platform.

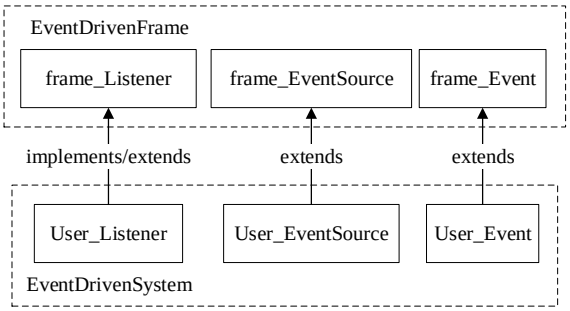


Figure 4. Event-driven pattern framework reuses behavior

3) Reasoning and inquiry

We reason and query based on ontology to match between the target system and the event-driven architecture, so as to obtain the event-driven architecture instance. We reason and query the model defined by Jena[7][8]. We use the ontology query function to obtain the instance of event-driven architecture in the extended ontology knowledgebase.

B. Refactor event-driven architecture

In this section, we refactor architecture based on event-driven architecture identification. As shown in Fig 5, we use the event-driven architecture violate specification to locate the refactoring points. Then we choose the corresponding refactoring scheme to eliminate or reduce the violation of the event-driven architecture, so as to obtain a new architecture. Repeat the steps until there is no violation of event-driven architecture in the target system.

Step 1: We locate the refactoring point in the identification result of event-driven architecture. Refactoring point is the violate specification of event-driven architecture. The event-driven architecture is a distributed processing pattern composed of highly decoupled event listeners with single responsibility. Therefore, the most important usage specifications of the event-driven architecture are the single responsibility of the listener specification and the distributed processing specification. The single responsibility of the listener specification requires a listener to handle only one type of events. If a listener class handles multiple types of events, the change of one event

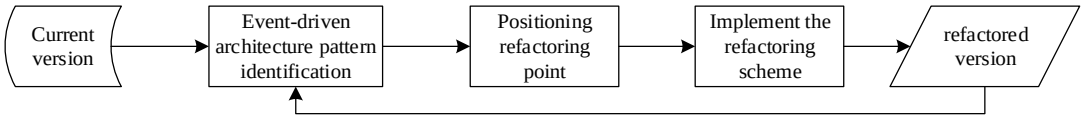


Figure 5. Refactoring process of event-driven architecture

handling method may weaken the handling ability of other events[9]. The distributed processing specification requires that the events are generated and processed in different classes. If a class is both an event source and a listener, it violates the distributed processing specification[5].

Step 2: we implement the scheme for the refactoring points. The appropriate refactoring scheme should specify the refactoring operation according to different refactoring points. In this paper, we propose two refactoring schemes, i.e., RS, for the refactoring points located by the single specification of listener responsibility and the distributed processing.

RS 1: The refactoring scheme for the single responsibility of the listener specification.

The refactoring scheme is proposed to eliminate the violation of the single responsibility of the listener specification. In this kind of violation, a listener class handles more than one type of events. The refactoring scheme is to split the listener class into several classes and let each class handle one type of events. As shown in Fig 6, we define a new empty listener class, and transfer the one of the events to the new class. At the same time, the corresponding dependencies are transferred.

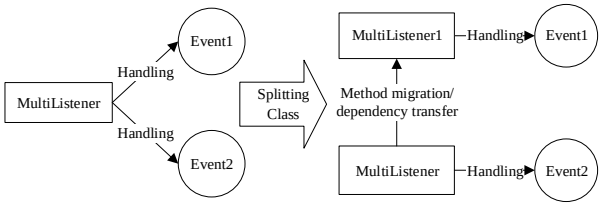


Figure 6. Split listener classes that handle various types of events.

RS 2: The refactoring scheme for distributed processing specification

The refactoring scheme is proposed to eliminate the violation of the regulations of the distributed processing specification. In this kind of violation, the class is both an event source and a listener. The refactoring scheme is to split the class into two classes, one class is listener and the other is event source.

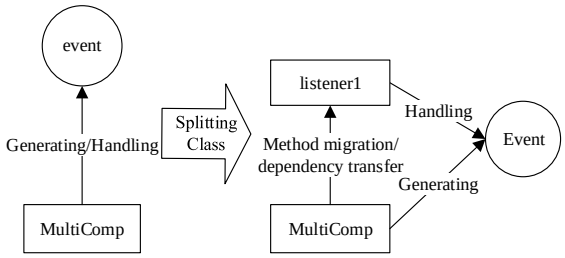


Figure 7. Splitting classes that are both event sources and listeners

As shown in Fig 7, we define a new empty listener class, and transfer the events handling to the new class. At the same time, the corresponding dependencies are transferred. The original class acts as an event source and the new class acts as a listener.

III. Experiment And Results

The identification and refactoring approach of event-driven architecture can identification and refactoring event-driven architecture based on ontology accuracy and efficiency, which can help developers understand and maintain software projects. In this section, we aim to answer the following research questions:

RQ1: How about the accuracy of the software architecture identification technique?

RQ2: How about the accuracy of the software architecture refactoring technique?

RQ3: How about the efficiency of the software architecture refactoring technique?

A. Experimental Setup

1) Subject projects

To answer the above research questions, we select 50 Java projects from GitHub and SourceForge according to the popularity of Java projects. These projects are more popularity with the key words, such as game, game engine, Java awt, Java swing and event-driven. We analysis the documents and source codes of these 50 projects manually to obtain the ground-truth architecture. We select freecol and shiro to analyze their refactoring points.

2) Measurement

We use Precision, Recall and Accuracy to measure the accuracy of software architecture identification technique. It is defined by the following formulas

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$A = \frac{TP + TN}{TP + FP} \tag{3}$$

Where P indicates Precision, R indicates Recall, A indicates Accuracy, TP, FN, FP and TN indicate four numerical values in the confusion matrix of identification results.

TABLE I. CONFUSION MATRIX OF IDENTIFICATION RESULTS

Manual analysis results	Identification result	
	Yes	No
Yes	*TP*	*FN*
No	*FP*	*TN*

We use Accuracy, CostRate and Effectiveness to measure the efficiency of the software architecture refactoring technique.

The Accuracy of refactoring point location is the proportion of correctly located refactoring points in all the refactoring points located by this technique.

The CostRate is the proportion of the number of classes to be refactored to the total number of classes in the object.

The Effectiveness is the proportion of eliminated refactoring points in all the refactoring points located by this technique.

3) Experimental *steps*

For each studied subject, we performed the following steps:

Step1: We collect 50 Java projects from GitHub and SourceForge with the key words.

Step2: For each selected project, we identify their architecture pattern manually to confirm wither they are event-driven architecture project.

Step3: For freecol and shiro, we obtain their ground-truth architecture manually to build the comparative experiments.

Step4: For freecol and shiro, we refactor their architecture base on ontology, and we collect all the results to analyze the accuracy and effectiveness.

Step5: For each event-driven architecture project, we obtain the refactoring points and process the refactoring schemes by manual analysis as reference, the effectiveness of the refactoring method based on event-driven architecture identification is evaluated through the accuracy of refactoring point positioning and refactoring cost rate.

In the experiments, we use computer with 64-bit Windows 10 and 8G memory. We use JDK1.8, Eclipse Neon 4.6.0, and MySql 5.6. The ontology inference engine witch we use is Jena 3.10.0.

B. Results analysis

RQ1: The accuracy of the software architecture identification technique

To evaluate accuracy of the event-driven architecture identification-based ontology, we apply the IRABO, and manual analysis work on the 50 projects. The manual analysis work of clone, freecol, jmonkeyengine, Jadventure, libgdx, AndEngine, overlap2d, GameHelper and Shiro is based on the source code and documents. The other 41 projects can only be analyzed according to the source code because of missing documents. Table II presents the results, "√" means the project is event-driven architecture, "×" means the project isn't event-driven architecture.

From Table III, we can find that there are 13 projects with event-driven architecture by manual analysis. There are 12 projects with event-driven architecture by IRABO identification. There are 8 projects whose manual analysis and RABO identification results are both event-driven architectures. The Precision, recall and accuracy of IRABO are 66.6%, 61.54% and 82%. There are 18% identification error rate of IRABO.

The reason of identification error rate of IRABO is false negative and false positive. The reason of false negative is as follows:

IRABO only considers the typical event-driven architecture when identifying the architecture, but it fails to identify the project with atypical event-driven architecture.

IRABO only considers the mainstream event-driven framework when identifying the architecture, but it fails to identify the non-mainstream event-driven framework.

TABLE II. IDENTIFICATION RESULTS

Project	IRABO	Manual analysis	Project	IRABO	Manual analysis	Project	IRABO	Manual analysis
clone	√	√	Terasology	×	×	blog	×	×
openbbs	×	×	pixel-dungeon	×	×	jnativehook	×	×
MyBlog	×	×	FunGameRefresh	×	×	jmonkeyengine	√	√
freecol	√	√	WorldEdit	×	×	Jadventure	×	√
terrier	×	×	JustWeEngine	×	√	jadx	×	×
lionengine	√	√	overlap2d	√	×	JHotDraw	×	×
junit4	×	×	StormPlane	×	×	libgdx	×	√
la4j	√	×	OpenRTS	√	√	AndEngine	×	√
okhttp	×	×	PretendYoureXyzzy	√	×	HikariCP	×	×
mybatis	×	×	Essentials	×	×	arthas	×	×
vert.x	×	×	GameHelper	×	×	Mosby	×	×
beautyeye	√	×	druid	×	×	latexdraw	×	×
symphony	×	×	SSH-master	×	×	MARIO	×	×
mockito	×	×	ssm-master	×	×	log4j	×	×
junit5	×	×	Examination_System	×	×	FXGL	×	×
litiengine	√	√	shiro	√	√	JabRef	×	×
inxedu	×	×	realm	×	×			

IRABO describes the event-driven architecture by describing the structural behavior characteristics of the observer pattern and its application. Therefore, the false identification of the observer pattern will lead to the false identification of the event-driven architecture.

TABLE III. EXPERIMENTAL RESULTS

Manual analysis	IRABO		
	Yes	No	Total
Yes	8	5	13
No	4	33	37
Total	12	38	50
Precision	66.67%		
Recall	61.54%		
Accuracy	82%		

The reason of false positives is as follows:

Architecture is the overall design of software. When the project partially implements the event-driven mechanism, IRABO would identify it as an event-driven architecture project.

We obtain the ground-truth architecture by manual analysis, and the false of manual analysis results will lead to false positives.

RQ2: The accuracy of the software architecture refactoring technique

We choose two typical event-driven architecture projects, freecol and shiro. We use IRABO to obtain the event-driven architecture instances of two projects compare with manual analysis results. The accuracy of IRABO is measured by Precision and Recall, as shown in Table IV.

TABLE IV. THE ACCURACY OF IRABO

Project	Component	*Precision*	*Recall*
freecol	audio monitor	75.45%	65.76%
	event	54.23%	44.54%
	Event source	58.51%	41.71%
shiro	audio monitor	83.35%	66.23%
	event	57.68%	46.54%
	Event source	65.43%	58.92%
Average value		65.78%	53.95%

From Table IV, we can find that the average Precision and Recall of IRABO are 65.78% and 53.95%. The false positives and false negatives in the event-driven architecture identification results are caused by event-driven architecture variants and false manual analysis result.

RQ3:The efficiency of the software architecture refactoring technique

a) Eliminate the single responsibility of listener specification violation

We positioned refactoring points that violate the single specification of listener responsibilities in all the event-driven architecture. We fined refactoring points in clone, freecol, lionengine and litiengine by IRABO and manual analysis. The Refactoring points positioned by IRABO and manual analysis work are shown in Table V. In clone, freecol, lionengine and litiengine, the Accuracy of IRABO is 50%, 39.1%, 46.2% and 60.2%, and the CostRate of IRABO is 0.035, 0.153, 0.172 and 0.272.

TABLE V. REFACTORING POINTS

Project	Classes	Refactoring points (IRABO)	Refactoring points (Manual)	Classes needing refactoring
clone	115	2	1	4
freecol	1224	192	75	188
lionengine	843	132	61	145
litiengine	445	88	53	121

We choose a refactoring point CanvasMouseListener in freecol, which violates the single specification of the listener specification. Then we refactor CanvasMouseListener by RS 1 to eliminate the single responsibility of listener specification violation.

b) Eliminate the distributed processing specification verification

We positioned refactoring points that violate the distributed processing specification in all the event-driven architecture. We fined refactoring points in reecol, lionengine and litiengine by IRABO and manual analysis. The Refactoring points positioned by IRABO and manual analysis work are shown in Table VI. In reecol, lionengine and litiengine, the Accuracy of IRABO is 75%, 65.2% and 77.3%, and the CostRate of IRABO is 0.009, 0.018 and 0.038.

TABLE VI. REFACTORING POINTS

Project	Classes	Refactoring points (IRABO)	Refactoring points (Manual)	Classes needing refactoring
freecol	1224	12	9	11
lionengine	843	23	15	15
litiengine	445	22	17	17

We choose a refactoring point BuildingPanel in freecol, which violates the distributed processing specification. Then we refactor BuildingPanel by RS 2 to eliminate the distributed processing specification verification.

IV. THREATS TO VALIDITY

Threats to external validity. The ground-truth architecture obtained by manual analysis is used to verify the accuracy of the architecture obtained by IRABO. Influenced by the ability of analysts or the complexity and scale of the project, the architecture obtained by manual analysis is subjective to some extent. That may threaten the accuracy of the software architecture identification and refactoring technique. To reduce this threat, we will select more excellent open-source projects of event-driven architectures; conduct a more comprehensive analysis to obtain ground-truth architecture more accurately.

Threats to internal validity. IRABO only considers the typical event-driven architecture when identifying the architecture. For the projects with atypical event-driven architecture, false negative and false positive may occur. To reduce this threat, we will consider more variants of event-driven architecture to build a more complete ontology knowledge base of event-driven architecture.

Limited by manpower and time, the projects selected in this paper are small-scale, which cannot verify the accuracy and effectiveness of this technique in large-scale projects. In the future work, we will repeat the experiments with more large-scale projects to reduce this threat.

V. Related Work

In the aspect of pattern identification and description of architecture, Mavridou Anastasia points out that architecture can be represented by logic and architecture style can be described by configuration[9]. Cortella Essav and others proposed to use logical predicates to model anti-patterns, and build an engine based on these logical predicates to detect anti-patterns in the target system[10]. Rabiaz et al. proposed a method of knowledge retrieval to identify instances of architecture patterns in software systems.[11]. The powerful ability of ontology description is exactly what is needed to describe the very high level of abstraction such as architectural patterns.[12]. Velasco-Elizondo P and others put forward an automatic analysis architecture model based on knowledge representation and information extraction, and then reconstructed the system according to the analysis results.[13]. The main problem of the existing architecture pattern identification and refactoring methods is the lack of a special method for event-driven architecture identification and refactoring. Therefore, this paper proposes an ontology-based pattern identification and refactoring method for event-driven architectures.

VI. Conclusion

In this paper, we present an approach of identification and refactoring approach of event-driven architecture based on ontology, i.e., IRABO. We identifying event-driven architecture based on ontology. We refactor event-driven architecture according to the usage specification of event-driven architecture. Experiments verify the accuracy of pattern identification based on ontology-based event-driven architecture and the effectiveness of the refactoring scheme. We evaluate IRABO by conducting experiments on 50 projects and compare with manual analysis work. The results show that IRABO perform efficiency and effectively. And there is still space for improvement of architecture recovery effectiveness. The follow-up work can start with the method of identification more variants of event-driven architecture to further improve the accuracy and effectiveness of software architecture recovery.

References

[1] Ta'id Holmes, and U. Zdun . Refactoring Architecture Models for Compliance with Custom Requirements[C]. ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems ACM, 2018.

[2] Elish K O , Alshayeb M . Using Software Quality Attributes to Classify Refactoring to Patterns[J]. Journal of Software, 2012, 7(2):p.408-419.

[3] Overbeek S , Janssen M , Bommel P V . Designing, formalizing, and evaluating a flexible architecture for integrated service delivery: combining event-driven and service-oriented architectures[J]. Service Oriented Computing&Applications, 2012, 6(3):167-188.

[4] Tragatschnig S , Stevanetic S , Zdun U .Supporting the evolution of event-driven service-oriented architectures using change patterns[J]. Information and Software Technology, (2018):133-146.

[5] Woodside M . Performance Models of Event-Driven Architectures[C]. CPE '21: ACM/SPEC International Conference on Performance Engineering ACM, 2021.

[6] Abel Gómez, Iglesias-Urkia M , Urbieta A , et al. A model-based approach for developing event-driven architectures with AsyncAPI[C].MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems. ACM, 2020.

[7] Yu Lei, Ma Hui, Wang Cheng. Research on equipment PHM knowledge ontology construction and semantic reasoning method[J]. ournal of Ordnance Equipment Engineering, 2019,40(S1):126-130.

[8] Lerlertvanich R, Vatanawood W. Facade Layer for Apache JENA[J]. Arpn Journal of Systems & Software, 2012, 2(11).

[9] A. Mavridou, E. Baranov, S. Bliudze, et al. Configuration logics: Modeling architecture styles[J]. Journal of Logical and Algebraic Methods in Programming, 2017, 86(1): 2-29.

[10] V. Cortellessa, A. D. Marco, C. Trubiani. An approach for modeling and detecting software performance antipatterns based on first-order logics[J]. Software & Systems Modeling, 2014, 13(1): 391-432.

[11] Rabinia Z , Moaven S , Habibi J . Towards a knowledge-based approach for creating software architecture patterns ontology[C].International Conference on Engineering & Mis. IEEE, 2016.

[12] Guessi M, Moreira D A, Abdalla G, et al. OntolAD: a Formal Ontology for Architectural Descriptions[C].ACM SAC 2015. ACM, 2015.

[13] Velasco-Elizondo P, Marín-Piña R, Vazquez-Reyes S, et al. Knowledge representation and information extraction for analysing architectural patterns[J]. Science of Computer Programming, 2016, 121: 176-189.

Designing Microservice-Oriented Application Frameworks

Yunhyeok Lee
Dept. of Computer & Information Science
University of Massachusetts Dartmouth
Dartmouth, Massachusetts, U.S.A.
ylee7@umassd.edu

Yi Liu
Dept. of Computer & Information Science
University of Massachusetts Dartmouth
Dartmouth, Massachusetts, U.S.A.
yliu11@umassd.edu

***Abstract*—An application framework is a reusable "skeleton" application that can be specialized to develop custom applications. As microservices become more popular in the software industry, the importance of methodologies for designing microservices-oriented frameworks is increasing. However, the existing methodologies for designing application frameworks, which were developed for the monolithic environment, do not fit the microservice architecture well. To address this issue, this study extends Schmid's systematic generalization approach to enable the design of microservice-oriented application frameworks. The methodology introduces communication spots, which define the execution orders among services, and communication styles, which define the communication relationship among services, to capture the communication characteristics in microservice environments. The methodology also proposes the strategies in microservice generalization and the usage of integration patterns in transforming a microservice-oriented application into a microservice-oriented framework. The new methodology can facilitate a developer to systematically design a microservice-oriented application framework by generalizing from a microservice application. A case study is used to demonstrate how to apply the proposed methodology to design a microservice fraud-detection framework.**

***Keywords*—software frameworks, microservice-oriented, design methodology**

I. Introduction

An application framework is "a generic application that allows different applications to be created from a family of applications." [18] Application frameworks facilitate the developers to reuse the framework's software artifacts to efficiently construct customized applications in particular business domains. An application framework contains the common aspects (*frozen spots*) that all family members have and variable aspects (*hot spots*) that vary among the application in the family [18]. The *hot spots* allow the framework to be customized to a specific member of the family.

The design of an application framework can be much more complicated than the design of a single application because the framework needs to address both common aspects and variable aspects in the application domain. Thus, the framework design requires a systematic approach.

Schmid[18] proposed a methodology of designing application frameworks using systematic generalization. The methodology examines the design of an existing application, identifies the frozen spots and hot spots in the family, and generalizes the application structure to construct a framework [6]. Built on the assumption of using the monolithic environment (single code-base and one-time deployment), the methodology is well suitable for designing object-oriented monolithic frameworks in which the components are designed to communicate based on the class relationships, such as inheritance, polymorphism, and composition. Microservice architecture [8] distinguishes itself from monolithic architectures by characteristics, such as modularity, fine grained services and communication mechanisms. In a microservice-oriented application, each service has its own process, which serves a single purpose and communicates with other services through lightweight mechanisms, such as APIs. As microservices are increasingly used in the development world, microservice-oriented application frameworks will be beneficial for the development of microservice-oriented applications in the particular business domains that the frameworks are constructed for. However, few study has been published on the methodology of designing microservice-oriented application frameworks. Schmid's approach is applicable for generalizing the hot spots and frozen spots within a microservice, but it is not sufficient to address the inter-microservices communications.

This research aims to propose a methodology for designing microservice-oriented application frameworks. The proposed approach is built on Schmid's systematic generalization methodology and extends it to fit into the microservices environment.

The rest of the paper is organized as follows. Section II briefly introduces the microservice architecture, Schmid's application framework design methodology and microservices integration patterns. Section III presents a methodology for designing microservice-oriented application frameworks. Section IV uses a case study to present how to apply the proposed methodology in designing a microservice fraud detection framework. Section V discusses related works similar to our approach, and Section VI concludes the work.

II. Background

This section provides a brief introduction to the microservices architecture and microservices integration patterns as

DOI reference number: 10.18293/SEKE2022-163

well as an overview of Schmid's systematic framework design from which our study extends.

A. Microservices Architecture

Martin Fowler defined the term "Microservices" [8] as a method to build a software application with a set of small services. Each service has its own process, which serves a single purpose and communicates with other services through application programming interfaces (API) [8]. *Modularity, fine-grained services,* and *simple communication mechanism* are the three key characteristics of microservices.

A microservice-oriented application contains a suite of independent modules in a system. Each of thoese modules, also called a microservice, encapsulates its domain logic and contributes to the overall functionality of its system, unlike the monolith which puts all the functionalities in a single process [8]. This modularity improves the flexibility of the development and the deployment of each service and increases the overall comprehensibility of the system.

The size of each microservice is comparatively small. Each service should focus on a single business capability to support low coupling within the system. The independent fine-grained services allow for a low cost of system maintenance and evolution into the future.

The microservices architecture focuses on lightweight communication mechanisms instead of hiding complexities in the communications. The HTTP request-response with resource APIs and lightweight messaging are commonly used in microservices to provide "dumb pipes" [8]. A simple communication approach enables changes in services without modifying the central service communication bus and offers the system better scalability to the overall system [3].

B. Microservice Integration Patterns

Microservices must be properly composed for any given system to function. There could be a large number of services in a system. In addition, a service may be reused within different scopes, which will increase the complexity of that service's composition. Service integration is crucial and challenging in the field of microservices architecture.

The case study of this research adopts the general service integration patterns [19], which are based on the interservice communication mechanisms within microservices. These patterns are *Synchronous Messaging*, *Asynchronous Messaging*, and *Hybrid Messaging*.

1) Synchronous Messaging Pattern: The communication between microservices is synchronous if microservice A sends microservice B a request that requires a response and A is blocked while waiting for said response [16]. The *Synchronous Messaging* pattern is designed to integrate microservices when they communicate via synchronous messaging.

2) Asynchronous Messaging Pattern: The communication is asynchronous if microservice A sends a request but is not blocked while waiting for a response if there is any [16]. The *Asynchronous Messaging* pattern is designed to integrate microservices that communicate via asynchronous messaging.

3) Hybrid Messaging Pattern: To realize a business need, a combination of synchronous and asynchronous messaging is typically required within a system. The *Hybrid Messaging* pattern provides a solution by combining aspects of the *Synchronous Messaging* pattern and the *Asynchronous Messaging* pattern.

C. Systematic Framework Design

Schmid's methodology of the systematic construction of an application framework consists of four steps [18]:

1) Creation of a Fixed Application Model: Schmid's approach starts by constructing a fixed application model with an object-oriented design for a specific application within the family.

2) Hot Spot Analysis and Specification: Once a complete model exists, the framework designer analyzes the model and domain to discover and specify potential hot spots.

3) Hot Spot High-Level Design: The features of any hot spots are accessed through the common interface of the abstract class. However, the design of the hot spot subsystem enables different concrete subclasses of the base class to be used to provide variant behaviors.

4) Generalization Transformation: The approach seeks to generalize the design around these hot spots by applying systematic transformations of the design that are driven by the analysis of the hot spot.

III. Methodology

The proposed methodology uses Schmid's approach as a basis and extends it to fit into the microservice-oriented environment. The methodology consists of six steps as described below.

A. Step 1. Develop a Solid Microservice-Oriented Application

Just like Schmid's approach, the proposed methodology starts with the construction of a representative application in the framework family. However, this application should be microservice-oriented, unlike the object-oriented, monolithic application in Schmid's approach.

B. Step 2. Identify Hot Spots and Frozen Spots

Aspects that vary among application family members are hot spots [18]. The different implementations to the hot spots result in different applications within the family. A framework is customized to a specific application with the specific implementations to the hot spots. The aspects that are common to all the application family members are called frozen spots [18]. These frozen spots are the basis of designing the overall structure of the framework and are fixed and reusable for all the applications within the family.

When designing a microservice-oriented framework, any hot spots and frozen spots should be identified for each service. Commonality and variability analysis approaches [5, 12] are beneficial for identifying the hot spots and frozen spots.

C. Step 3. Analyze and Specify Hot Spots

We adopt the Schmid's approach in analyzing the high-level hot spots and specifying the details [18]. All identified hot spots from the domain are collected and evaluated. Each hot spot is specified by a short description of its purpose, its common responsibility, the kinds of variability and the multiplicity.

D. Step 4. Design Hot Spot Subsystems

A hot spot is captured within the scope of a microservice and implemented by a hot spot subsystem. Schmid's strategy of designing hot spot subsystems is adopted for this phase. A hot spot subsystem consists of an (abstract) interface defining the common responsibilities, concrete implementations (for addressing variability) to the interface, and additional classes and relationships [18]. Design patterns [17] are beneficial for determining the structures of the hot spot subsystems on capturing the abstract interface and concrete implementations and their relations.

E. Step 5. Identify Communication Spots and Communication Styles

In a monolithic object-oriented application, communications among the components are typically conducted through the procedure calls via inheritance, polymorphism, or composition. Such communication is applicable for the components within a microservice, but would not work between the microservices.

We use *communication spots* and *communication styles* to specify how the microservices communicate within a system. We define a *communication spot* as the execution order between two services. For example, microservice A executes prior to microservice B.

A *communication style* is defined as the procedure between the server and client to communicate when there is a request from the client side and the server side is expected to respond to the client's request. We use communication styles to describe the communications between microserivces or between clients and microservices.

The communication styles are categorized as synchronous and asynchronous, where the synchronous method is that a client side sends a request to a server-side service and waits for its response, and the synchronous method is that a client-side keeps sending requests to the server-side service without waiting for the acknowledgment of the previous response [11].

We use the notation $A(C, S)$ to describe that client-side C communicates with server-side service S through the asynchronous style. We use $S(C, S)$ to describe that client-side C communicates with server-side service S through the synchronous style.

This stage produces two descriptions: the *communication spot description* that consists of a list of the execution order of each pair of services, and the *communication style description* that contains a list of each pair of client-side and server-side service's communication style.

F. Step 6. Transform into a Microservice-Oriented Framework

Schmid's generalization transformation strategy uses the object-oriented techniques, such as inheritance, polymorphism, and composition, to generalize the design around the hot spots for constructing the final framework structure. Such transformation strategy is applicable within each microservice. Beyond the structure of each service, a microservice-oriented framework also needs to reflect the microservice generalization and service communications.

We propose two options in generalizing the microservices. The first option, as shown in Fig. 1, captures the abstract interface and concrete implementations to the interface within a microservice; while the second option, as shown in Fig. 2, captures the abstract microservices in the framework only.

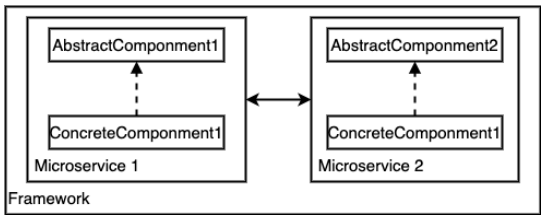


Fig. 1. Microservice Generalization - Option 1

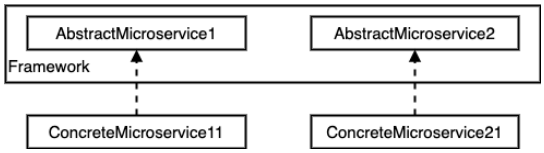


Fig. 2. Microservice Generalization - Option 2

The microservices should be integrated properly in the framework. The identified communication spots and communication styles determine the final integration transformation. The three integration patterns (Synchronous Messaging, Asynchronous Messaging, and Hybrid Messaging), described in Section II.B, can facilitate the integration process.

IV. Case Study

In this section, the design of a fraud detection application framework is used to illustrate how the proposed approach can be applied.

A. Overview of a Fraud Detection Application

Inspired by the fraud detection analysis on fraudulent credit card transactions [15], we have developed a prototype microservice application that uses machine-learning techniques to predict fraudulent credit card transactions. The application consists of four microservices, which are described below:

- *Dataset Uploading service* is for users to upload a dataset to the application. The CSV file format is accepted by default; however, the service converts a non-CSV files (such as XLSX) to CSV format.
- *Preprocessing service* normalizes data and selects important features for further analysis.

- *Fraud Detection service* applies the machine learning algorithms, such as Random Forest [21], Support Vector Machine [22], and Logistic Regression [4], etc., to detect fraud cases in the dataset.
- *Evaluation service* presents the results from *Fraud detection* as tables and heatmaps.

Figure 3(a) shows the user interface of the prototype application, while 3(b) and (c) illustrate the results by applying the *Random Forest* algorithm in a heatmap and a table.

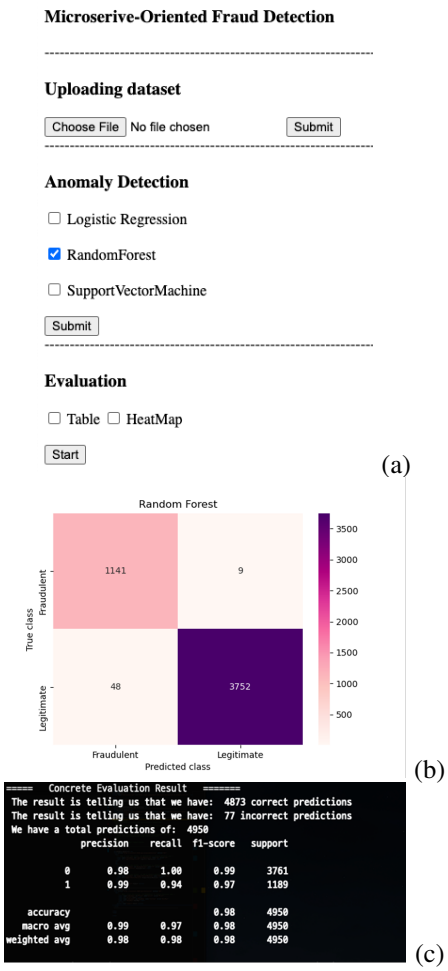


Fig. 3. Prototype Fraud Detection System

In this research, the scope of the fraud detection application family is constrained to analyze only stationary datasets on classical computers using CPUs.

B. Design of Fraud Detection Framework

In order to allow the developers to design their own user interfaces (UIs), the fraud detection framework does not include UI components, but provides interfaces to connect to UIs.

Since a specific microservice application has already been built, the design of the fraud detection framework begins with step 2 of the design approach.

1) Identification of Frozen Spots and Hot Spots: By analyzing the design decisions and the scope of the application family, we choose the following frozen spots:

- *Frozen Spot 1.* The data input and analysis order is fixed, that it, dataset upload, data preprocessing, fraud analysis and detection, and presentation of results.
- *Frozen Spot 2.* A dataset is uploaded one at a time and then converted to CSV format.
- *Frozen Spot 3.* Machine learning algorithms are used in the *Fraud Detection* service.
- *Frozen Spot 4.* Users should be able to upload a dataset, choose machine learning algorithms and view the results.

After examining the scope and the prototype application, we identify the following aspects that vary among applications in the fraud detection family, thus define them as hot spots:

- *Hot Spot 1.* Variability in the file formats of datasets uploaded by users in the *Dataset Uploading* service. In addition to the default CSV format, other file formats should also be supported, such as XLXS, mat, txt, etc.
- *Hot Spot 2.* Variability in the data normalization techniques in the *Preprocessing service*.
  Data normalization can be done using various methods, such as decimal scaling, min-max normalization, etc. Some text-based datasets may need natural language processing [13] before further analysis.
- *Hot Spot 3.* Variability in the feature selection techniques in the *Preprocessing* service.
  Various methods can be applied to select the important features such as Recursive Feature Elimination [2], LASSO [7], etc.
- *Hot Spot 4.* Variability in the machine learning algorithms applied to analyze a dataset in *Fraud Detection* service.
  Various machine learning algorithms is applied to analyze a dataset. These algorithms include supervised learning (e.g. Random Forest) and semi-supervised learning (e.g. Convolutional Neural Network [1]).
- *Hot Spot 5.* Variability in the result presentation in the *Evaluation* service.
  In addition to the heatmap and table presentation implemented in the prototype, other data visualization techniques can be added, such as AUROC curve [14].

2) Hot Spot High-Level Specification and Design: This stage of the design involves the detailed specification of each identified hot spot and the design of the hot spot subsystems.

Due to the page limit of this paper, we only present the specification of hot spot 4, and its hot spot subsystem design.

Hot Spot 4: Variability in the machine learning algorithms applied to analyze a dataset in *Fraud Detection* service.

- Description: To allow a variety of machine learning algorithms.
- Common responsibility is to perform fraud detection on a dataset.
- The kinds of variability required are the algorithms in the categories of unsupervised learning, supervised learning, and semi-supervised learning.

- The multiplicity is one since one machine learning algorithm is used at a time.

The various machine learning algorithms are considered to be interchangeable in the fraud detection. The subsystem of hot spot 4 should allow new algorithms to be added and existing algorithms to be modified or removed without impacting the remaining parts of the system. In addition, a convenient way to access each algorithm should be provided. The *Strategy* design pattern defines "a family of algorithms, encapsulates each one, and makes them interchangeable; Strategy lets the algorithm vary independently from clients that use it." [9] This pattern is the best fit for organizing and managing the independent machine learning algorithms.

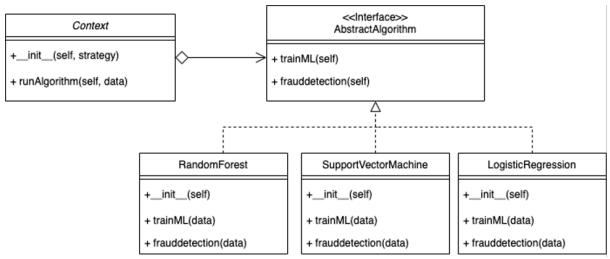


Fig. 4. Hot Spot Subsystem of Fraud Detection

As illustrated in Fig. 4, the interface *AbstractAlgorithm* defines the methods that are common in the machine learning algorithms which are supported in this application family; A machine learning algorithm, such as *Random Forest, Support Vector Machine, Logistic Regression*, etc., is implemented as a concrete class to the *AbstractAlgorithm* interface; The *Context* maintains a reference to each machine learning algorithm object and forwards the requests from its clients to the machine learning algorithm objects.

3) Identification of Communication Spots and Communication Styles: The execution order of a pair of services that communicate with one another is identified as a communication spot. By examining the framework's scope, we can capture the following communication spots:

- *Communication Spot 1*: The *Dataset Uploading* service executes prior to the *Preprocessing* service.
- *Communication Spot 2*: The *Preprocessing* service executes prior to the *Fraud Detection* service.
- *Communication Spot 3*: The *Fraud Detection* service executes prior to the *Evaluation* service.

The *Dataset Uploading* service communicates with a client through the synchronous communication method. A client sends a synchronous request to the *Dataset Uploading* service.

We analyze the communication styles by inspecting how each serve-side service responds to a client-side's request. The identified communication styles are summarized below:

- S(client, Dataset Uploading): A client sends a synchronous request to the *Dataset Uploading* service. Only a single file can be uploaded at a time.
- S(client, Preprocessing): A client sends a synchronous request to the *Preprocessing* service. One a single dataset is preprocessed at a time.
- A(Preprocessing, Fraud Detection): A client-side *Preprocessing* sends asynchronous requests to server-side *Fraud Detection* service. Multiple machine learning algorithms can be run concurrently.
- S(client, Evaluation): A client sends a synchronous request to *Evaluation* service. The result of one dataset is presented at a time.

4) Transformation to a Microservice-Oriented Framework: We adopt the first component organization approach (Fig. 1) to plug in the hot spot subsystems to the framework. For example, the Fraud Detection strategy (Fig. 4) along with its concrete implementations (*Random Forest, Support Vector Machine, Logistic Regression*, etc. are wrapped in the *Fraud Detection* service.

The identified communication styles indicate that the services use a mix of synchronous and asynchronous methods, thus, the *Hybrid Messaging* pattern is the best to be applied to integrate the services. With the help of the identified communication spots for the communication orders of the services, the high-level integration structure of the Fraud Detection framework is designed as shown in Fig. 5.

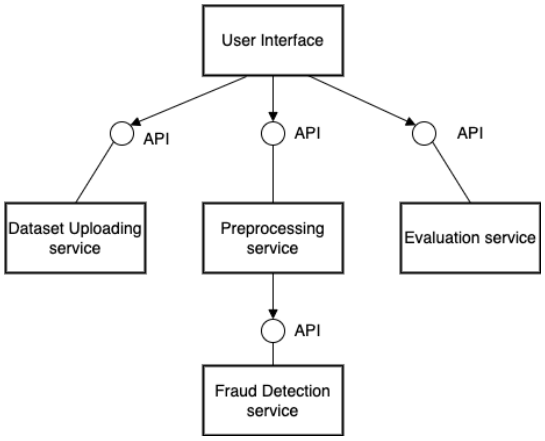


Fig. 5. Applying Hybrid Messaging Pattern to Integrate Services in Fraud Detection Framework

V. Discussion

Schmid's approach is object-oriented, generalizing the class structure of an application when designing application frameworks. There are also alternative systematic approaches that are not constrained to the object-oriented paradigm. Functional generalization is an example [6]. The function generalization approach generalizes the functional structure of an executable specification to produce an application framework. Hot spots are introduced to the design by replacing concrete operations with general abstract operations. These abstract operations become parameters of the generalized functions. Our proposed methodology can adopt such an alternative systematic

approach by modifying step 4 and 6 to generalize functions instead of dealing with classes.

This study uses three integration patterns in the case study. There are also other alternative integration patterns that can be used in the transformation step. For example, Microsoft Azure proposed nine patterns [20] for integrating microservices in applications. Each pattern serves as a solution for a fine-grained integration problem. To construct a system with microservices, it is necessary to utilize most of these patterns simultaneously. Gupta [10] introduced six patterns to provide multiple integration approaches from varying perspectives.

VI. Conclusion

Microservice-oriented application frameworks will be beneficial for the development of microservice-oriented applications in the particular business domains that the frameworks are constructed for. The existing design approaches for application framework design are limited to a monolithic environment that does not fit into the microservice architecture. This research proposes a methodology that extends Schmid's systematic generalization methodology by introducing communication spots and transformation strategies that fit in the microservice architecture. A case study is used to demonstrate the usage of the proposed methodology in designing a microservice-oriented fraud detection framework. Future work will be focused on two directions. One direction is to enhance the fraud detection framework by introducing the support for time series, unsupervised learning algorithms, and the usage of quantum computing. The other direction is to examine the impact of the different computing environments (classical vs. quantum) on the framework design thus enhance the methodology. The enhancement from the first direction will set a practical basis for the study of the second direction.

References

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. "Understanding of a Convolutional Neural Network". In: *2017 International Conference on Engineering and Technology (ICET)* (2017), pp. 1–6.

[2] Brandon D. Butcher and Brian J. Smith. "Feature Engineering and Selection: A Practical Approach for Predictive Models". In: *The American Statistician* 74 (2020), pp. 308–309.

[3] T. Cerný, M. J. Donahoo, and J. Pechanec. "Disambiguation and Comparison of SOA, Microservices and Self-Contained Systems". In: *Proceedings of the International Conference on Research in Adaptive and Convergent Systems* (2017).

[4] Wenlin Chen et al. "Density-Based Logistic Regression". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013).

[5] James O. Coplien, Daniel Hoffman, and David M. Weiss. "Commonality and Variability in Software Engineering". In: *IEEE Softw.* 15 (1998), pp. 37–45.

[6] H. Conrad Cunningham, Yi Liu, and Pallavi Tadepalli. "Framework design using function generalization: a binary tree traversal case study". In: *ACM-SE 44*. 2006.

[7] Valeria Francesca Fonti and Eduard N. Belitser. "Feature Selection using LASSO". In: *VU Amsterdam Research Paper in Business Analytics* 30 (2017), pp. 1–25.

[8] M. Fowler. *Microservices: a Definition of This New Architectural Term*. 2014. URL: https://martinfowler.com/articles/microservices.html.

[9] Erich Gamma et al. "Design patterns: elements of reuseable object-oriented software". In: 1994.

[10] A. Gupta. *Microservice Design Patterns*. 2015. URL: https://uberconf.com/blog/arun_gupta/2015/04/microservice_design_patterns.

[11] Mingyu Lim. "Directly and Indirectly Synchronous Communication Mechanisms for Client-Server Systems Using Event-Based Asynchronous Communication Framework". In: *IEEE Access* 7 (2019), pp. 81969–81982.

[12] R AL-msie'Deen et al. "Detecting commonality and variability in use-case diagram variants". In: *ArXiv* abs/2203.00312 (2022).

[13] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association : JAMIA* 18 (2011), pp. 544–51.

[14] S. Narkhede. *Understanding AUC - ROC Curve*. 2018. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[15] R. Polantizer. *Fraud Detection in Python; Predict Fraudulent Credit Card Transactions*. 2021. URL: https://medium.com/@polanitzer/fraud-detection-in-python-predict-fraudulent-credit-card-transactions-73992335dd90.

[16] C. Richardson and F. Smith. *Microservices from design to deployment*. NGINX, Inc., 2016.

[17] H. A. Schmid. "Design Patterns for Constructing the Hot Spots of a Manufacturing Framework". In: *Journal Object Oriented Programming* 9 (1996), pp. 25–37.

[18] H. A. Schmid. "Systematic framework design by generalization". In: *Communication of ACM* 40 (1997), pp. 48–51.

[19] M. Wang. "Service Integration Design Patterns in Microservices". MA thesis. South Dakota State University, 2018.

[20] M. Wasson. *Microservices: a Definition of This New Architectural Term*. 2017. URL: https://azure.microsoft.com/en-us/blog/design-patterns-for-microservices/.

[21] Q. Wu et al. "ForesTexter: an Efficient Random Forest Algorithm for Imbalanced Text Categorization". In: *Knowledge-Based Systems* 67 (2014), pp. 105–116.

[22] Yue Zhang and Zhimeng Feng. "A SVM Method for Continuous Blood Pressure Estimation from a PPG Signal". In: *Proceedings of the 9th International Conference on Machine Learning and Computing* (2017).

Collaborative Web Service Quality Prediction via Network Biased Matrix Factorization

Wenhao Zhong, Yugen Du*, Chuang Shan, Hanting Wang, Fan Chen
Shanghai Key Laboratory of Trustworthy Computing
Software Engineering Institute, East China Normal University
Shanghai, China
ygdu@sei.ecnu.edu.cn

***Abstract*—Facing a large number of candidate Web service with the same function, user wishes to get the most appropriate one. Quality-of-Service (QoS) which represents non-functional attributes of Web services, has become a major concern for choosing service. But it is time-consuming and resource-consuming to assess all the QoS values by invoking candidate services one by one. Thus, QoS prediction is considered an effective method to obtain QoS information. Although most of QoS prediction methods claim be able to capture the interaction between users and services, few of them take account non-interaction factors, especially the factors arising from the network environment. In this paper, the non-interaction factors from the network environment are referred as network bias, and a network biased matrix factorization (NBMF) method is proposed for QoS prediction. The method packages network bias into a linear regression model and puts the user-service interaction into a matrix factorization model, which is more sophisticated in adapting diversified circumstance, particularly in complex network environment. In addition, extensive experiments are conduct on real-world QoS dataset, and the result prove that the NBMF method achieves better performance than other state-of-the-art methods.**

***Index Terms*—Web services, QoS prediction, matrix factorization, network bias**

I. Introduction

The statistics published by ProgrammableWeb[1] indicate rapid growth in the number of published Web services over the past few years. The popularity of Web services allows different service-oriented applications and systems to be built to meet the increasingly complex business requirements [1].

To ensure the performance of service-oriented applications and systems, the quality of their component Web services needs to be assured. The quality of Web services can be described by their functional and non-functional attributes. Quality-of-Service (QoS) represents non-functional attributes of Web services, such as response time, throughput, availability, and reliability [2]. Since there are many Web services with similar functions on the network, investigating non-functional QoS attributes becomes a major concern for service selection [3], [4]. In practice, it is not easy to obtain the QoS values of all candidate services. First, the QoS values observed by users depend heavily on the invocation environment, and the quality of the same web service observed by different users may be very different [5]. Second, it is time-consuming and resource-consuming to assess all the QoS values by invoking candidate services one by one, due to a large number of users and services [6], [7]. Therefore, QoS prediction has attracted the attention of many researchers over the past few years and is considered an effective way to obtain QoS information.

Matrix factorization (MF) is arguably the most popular model-based collaborative filtering technique for QoS prediction [8], [9]. MF attempts to capture the interaction between users and services [10], [11], which factorizes the high-rank user-service matrix into two low-rank feature matrices [1], [3], and the inner product of the feature matrices represents the predicted QoS values of services observed by users. In addition to the user-service interaction, there are many factors unrelated to the interaction in the real-world prediciton. Although Zhu et al. [10] propose using user bias and service bias to capture the non-interaction from users and services, they do not take into account the non-interaction from the network environment. In the case of response time, the user-perceived response time must include network latency [12], which can vary significantly depending on the network path.

In this paper, we refer to the non-interaction factors from the network environment as network bias, and propose a network biased matrix factorization (NBMF) method for QoS prediction. Our method considers both interaction and non-interaction between users and services. Among the factors related to the non-interaction, we focus on the influence of network bias on QoS prediction. Specifically, our method packages network bias into a linear regression model, while putting the user-service interaction into a matrix factorization model, which makes our method adaptable to complex network environment.

In summary, the main contributions of this paper include:

- We propose a network biased matrix factorization method for QoS prediction. Our method considers both interaction and non-interaction between users and services, which makes our method adaptable to complex network environment.
- We conduct extensive experiments on a real-world QoS dataset to evaluate the performance of our method. The result demonstrates that our method can achieve better performance than the state-of-the-art baseline methods.

[1]https://www.programmableweb.com/

DOI reference number: 10.18293/SEKE2022-113

The remainder of this paper is organized as follows. In Section II, we explain the motivation of this work. In Section III, we introduce the proposed method. In Section IV, we conduct experiments on a real-world dataset to illustrate the effectiveness of the proposed method. In Section V, we review work related to our method. Finally, we conclude our work with future directions in Section VI.

II. Motivation

The traditional MF may not produce personalized QoS prediction, because it ignores the non-interaction bias between users and services, especially the non-interaction bias from the network environment. To explain this problem in detail, this section first introduces MF-based QoS prediction and then gives an example to illustrate the influence of network bias on MF-based QoS prediction.

A. Matrix Factorization

MF uses a factor model to fit the historical invocation matrix for prediction, which factorizes the user-service matrix into two low-rank feature matrices [1], [3]. The low-rank feature matrix attempts to explain the QoS data by describing the values on various latent features (e.g., system structure, hardware composition, software configuration). Each row of the user feature matrix represents the latent feature of a user, each row of the service feature matrix represents the latent feature of a Web service, and the dot product of them represents the user-service interaction, that is, the QoS value of the service observed by the user. Thus, the general objective function for the MF-based QoS prediction method can be derived as:

$$\hat{Q}_{ij} = U_i W_j^T \tag{1}$$

Where $U \in \mathbb{R}_{m \times d}$ denotes the user latent feature matrix, and $W \in \mathbb{R}_{n \times d}$ denotes the service latent feature matrix. The vector $U_i (1 \leq i \leq m)$ denotes the latent feature vector of user i, and the vector $W_j (1 \leq j \leq n)$ denotes the latent feature vector of service j. The number of latent features in our model is d. The predicted QoS value of Web service j observed by user i is $\hat{Q}_{ij}$.

B. Motivating Example

To explain the influence of network bias on MF-based QoS prediction, a straight-away example is given in Fig. 1(a). In this example, we need to predict the QoS values of services w_1, w_2, w_1', w_2' observed by users u_1, u_2. We assume that u_1, u_2 are located in region R_1, w_1, w_2 are located in region R_2, and w_1', w_2' are located in region R_3. Where services w_1, w_2 have the same performance as services w_1', w_2', the average response time between R_1 and R_2 is 1s, and the average response time between R_1 and R_2 is 2s.

The QoS prediction of traditional matrix factorization is shown in Fig. 1(b), we assume that the QoS value of the service observed by the user is determined by two latent features, which are hardware composition and software configuration. In the following, we will focus on user u_1 and services w_1, w_1'. The user-perceived hardware composition feature to u_1 is 0.6, and the user-perceived software configuration feature to u_1 is 0.4. The service-provided hardware composition feature to w_1 is 1.0, and the service-provided software configuration feature to w_1 is 0.5. The dot product of u_1 and w_1 is 0.8, which represents the QoS value of w_1 observed by u_1. Since the performance of services w_1 and w_1' are exactly the same, they have the same degree of latent features, the QoS value of w_1' observed by u_1 is also 0.8.

However, the traditional matrix factorization ignores the non-interaction bias between users and services, especially the non-interaction bias from the network environment. In Fig. 1(a), although the performance of services w_1 and w_1' are the same, they belong to different regions (w_1 is located in R_2, w_1' is located in R_3), and their QoS values observed by u_1 should be very different. In this case, the QoS prediction made by previous work is not accurate.

III. Method

Since the non-interaction bias from the network environment accounts for a large proportion of the observed QoS values, it is crucial to model this bias accurately. To achieve this goal, we propose a network biased matrix factorization (NBMF) method for QoS prediction. In this section, we first introduce the proposed NBMF method, then give an example to illustrate the improved prediction, and finally describe the model training and parameter optimization process.

A. Network Biased Matrix Factorization

If we consider the network bias to be continuous rather than discrete, then linear regression can be used to predict the network bias between users and services. Following the above idea, we propose a network biased matrix factorization method. Our method packages network bias into a linear regression model, while putting the user-service interaction into a matrix factorization model. Thus, the general objective function for NBMF-based QoS prediction method can be derived as:

$$\hat{Q}_{ij} = \alpha(\mu_{xy} + \mathbf{b}_i + \mathbf{p}_j) + (1 - \alpha) U_i W_j^T \tag{2}$$

The first term $\alpha(\mu_{xy} + \mathbf{b}_i + \mathbf{p}_j)$ of the objective function is a linear regression model used to predict the network bias between user i and service j. Where, x is the network of user i, y is the network of service j, and μ_{xy} is the average QoS value between network x and network y. $\mathbf{b}_i (1 \leq i \leq m)$ denotes the bias between the QoS observed by user i and other users in the same network. $\mathbf{p}_j (1 \leq j \leq n)$ denotes the bias between the QoS provided by service j and other services in the same network.

The second term $(1 - \alpha) U_i W_j^T$ of the objective function is a matrix factorization model used to capture the interaction between user i and service j , where $U_i (1 \leq i \leq m)$ denotes the latent feature vector of user i, $W_j (1 \leq j \leq n)$ denotes the latent feature vector of service j, and their dot product $U_i W_j^T$ represents the interaction between user i and service j.

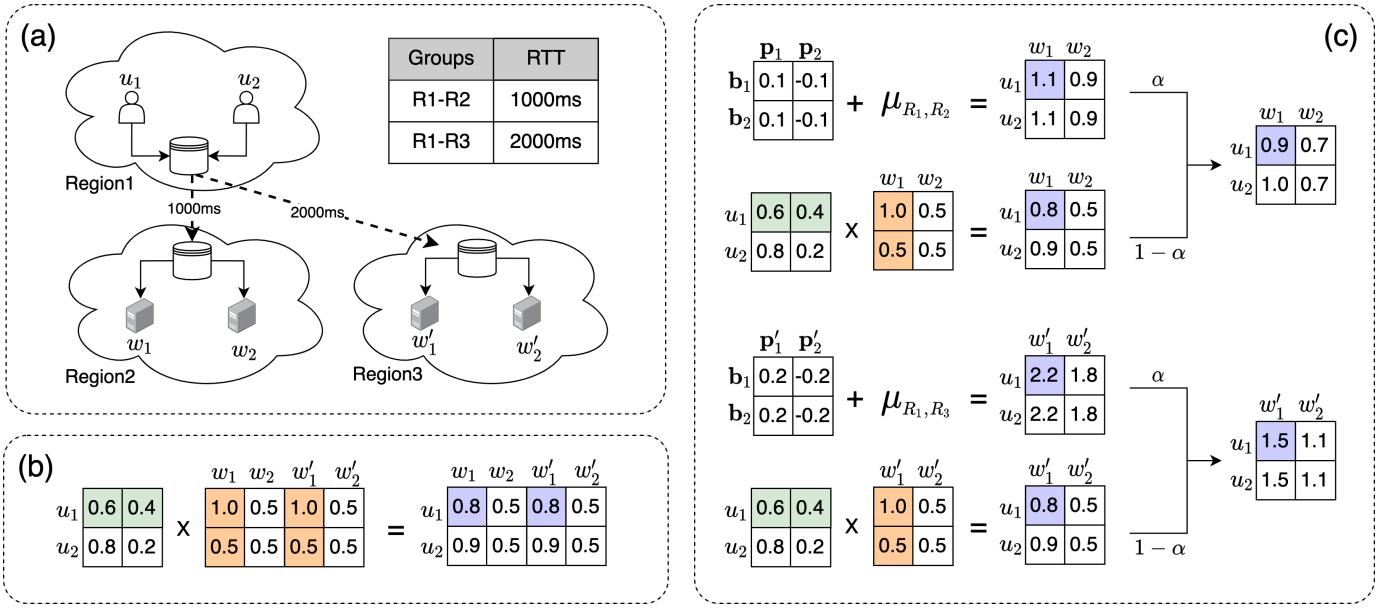


Fig. 1. An example to explain the influence of network bias on MF-based QoS prediction: (a) A real-world Web service invocation scenario; (b) MF-based QoS prediction; (c) NBMF-based QoS prediction.

The weight $\alpha(0 \leq \alpha \leq 1)$ measures the degree of network bias in our prediction model. α is an adjustable parameter. If α is set to 0, our prediction model does not consider network bias and only uses matrix factorization for prediction. If α is set to 1, our prediction model does not consider the user-service interaction and only uses linear regression for prediction. In order to investigate the impact of α on our model and find an optimal model, the value of α will be evaluated in Section IV.

B. Method Example

The improved QoS prediction is shown in Fig. 1(c). In the case of user u_1 invoking services w_1, w_1', the average QoS μ_{R_1,R_2} between region R_1 and region R_2 is 1, and the average QoS μ_{R_1,R_3} between region R_1 and region R_3 is 2. We assume the bias $\mathbf{b}_1$ between u_1 and other users in R_1 is 0, the bias $\mathbf{p}_1$ between w_1 and other services in R_2 is 0.1, the bias $\mathbf{p}_1'$ between w_1' and other services in R_3 is 0.2. Then the network bias between u_1 and w_1 is 1.1, and the network bias between u_1 and w_1' is 2.2. The next part of the QoS prediction is the user-service interaction, and the detailed procedure can be found in Section II-B.

The parameter α is set to 0.5 by default. After weighted sum of the network bias and the interaction, we can get the predicted QoS value of w_1 observed by u_1 is 0.9, and the predicted QoS value of w_1' observed by u_1 is 1.5.

Although the performance of services w_1 and w_1' are the same, they belong to different regions, and their QoS values observed by u_1 are different. It can be seen that our method is adaptable to complex network environment.

C. Model Training

The latent feature matrices and the bias vectors in Eq. (2) can be constructed by statistical learning theory. To estimate the values of matrices U, W and vectors $\mathbf{b}, \mathbf{p}$, we approximate the original matrix Q with the following objective function, and the minimization formula is as follows:

$$L = min_{U,W,\mathbf{b},\mathbf{p}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (Q_{ij} - \hat{Q}_{ij})^2 \tag{3}$$

Where I_{ij} is the indicator function that returns 1 if user i has invoked service j, and 0 otherwise. $\hat{Q}_{ij}$ is the prediction function as in Eq. (2). To avoid overfitting in approximating the original matrix, we add four regular terms related to U, W and $\mathbf{b}, \mathbf{p}$.

$$\begin{aligned} L = min_{U,W,\mathbf{b},\mathbf{p}} & \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (Q_{ij} - \hat{Q}_{ij})^2 \\ & + \frac{\lambda}{2} (\|U\|_F^2 + \|W\|_F^2 + \|\mathbf{b}\|_F^2 + \|\mathbf{p}\|_F^2) \end{aligned} \tag{4}$$

Where $\|\cdot\|$ denotes the *Frobenius norm* [13], the parameter λ controls the degree of regularization. The objective function based on L_2-norm as in Eq. (4) is not convex, it is unrealistic to design an algorithm to find the global minimum [14]. Instead, the stochastic gradient descent technique [15] can be employed to find the approximate optimal solution. For each QoS record observed when user i invokes service j, we have the following update rules:

$$U_i' = U_i - \eta \frac{\partial L}{\partial U_i} \tag{5}$$

$$W_j' = W_j - \eta \frac{\partial L}{\partial W_j} \tag{6}$$

$$\mathbf{b}_i' = \mathbf{b}_i - \eta \frac{\partial L}{\partial \mathbf{b}_i} \tag{7}$$

$$\mathbf{p}_j' = \mathbf{p}_j - \eta \frac{\partial L}{\partial \mathbf{p}_j} \tag{8}$$

where $\eta > 0$ represents the learning rate of updating the feature matrices and bias vectors, and

$$\frac{\partial L}{\partial U_i} = \lambda U_i - (Q_{ij} - \hat{Q}_{ij})(1-\alpha)W_j \tag{9}$$

$$\frac{\partial L}{\partial W_j} = \lambda W_j - (Q_{ij} - \hat{Q}_{ij})(1-\alpha)U_i \tag{10}$$

$$\frac{\partial L}{\partial \mathbf{b}_i} = \lambda \mathbf{b}_i - (Q_{ij} - \hat{Q}_{ij})\alpha \tag{11}$$

$$\frac{\partial L}{\partial \mathbf{p}_j} = \lambda \mathbf{p}_j - (Q_{ij} - \hat{Q}_{ij})\alpha \tag{12}$$

The overall optimization procedure of our method is given in Algorithm 1. Let r denote the number of iterations to achieve convergence, let s denote the number of valid invocation records in the original matrix Q, and let d denote the dimensions of the user latent feature matrix and the service latent feature matrix. The main time cost of Algorithm 1 lies in the updating of matrices U, W and vectors $\mathbf{b}, \mathbf{p}$. In each iteration, updating U, W takes $\mathcal{O}(sd)$ time and updating $\mathbf{b}, \mathbf{p}$ takes $\mathcal{O}(s)$ time. Thus, the overall time complexity of our method is $\mathcal{O}(rsd)$.

Algorithm 1 Optimization procedure of NBMF

Input: $Q \in \mathbb{R}^{m \times n}, \alpha, d, \lambda, \eta$;
Output: $U \in \mathbb{R}^{m \times d}, W \in \mathbb{R}^{n \times d}, \mathbf{b} \in \mathbb{R}^m, \mathbf{p} \in \mathbb{R}^n$

```
Randomly initialize U and W;
Initialize b and p with zero;
repeat
    for each record (i, j, Q_ij) observed in Q do
        Update U_i according to Eq. (5);
        Update W_j according to Eq. (6);
        Update b_i according to Eq. (7);
        Update p_j according to Eq. (8);
    end for
until Convergence
return U, W, b, p;
```

IV. Experiment

In this section, we conduct extensive experiments on a real-world QoS dataset to evaluate the performance of NBMF. The experiments are designed to address the following questions: (1) How does NBMF method compare with other state-of-the-art baseline methods? (2) How does the matrix density affect the prediction accuracy? (3) How does the weight α of the network bias affect the prediction accuracy? (4) How does the dimension d of the latent feature matrix affect the prediction accuracy? We implement our method and all baseline methods in Python 3.7, and all experiments were performed on a Linux server with Intel i5-10400 2.9GHz CPU and 16GB RAM running 64-bit Ubuntu 16.04.

A. Dataset

We conduct all experiments on a publicly real-world QoS dataset named WS-DREAM[2]. The dataset includes 1,974,675 QoS records, which were collected from 339 users in 30 regions on 5825 Web services in 73 regions. There is a QoS record between each user and each service, and we focus on the response time (RT) in the QoS attribute. Also, the dataset collects the IP, region and other information of these users and services. More details about this dataset can be found in [16].

B. Evaluation Metrics

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics are used to measure the accuracy of prediction by calculating the difference between the predicted QoS value and the actual QoS value.

MAE is defined as:

$$MAE = \frac{1}{N} \sum_{i,j} |Q_{ij} - \hat{Q}_{ij}| \tag{13}$$

RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i,j} (Q_{ij} - \hat{Q}_{ij})^2} \tag{14}$$

Where Q_{ij} and $\hat{Q}_{ij}$ represent the actual and predicted QoS value of the user i invoking service j, and N denotes the number of predicted QoS values. It can be observed from the formula that RMSE is more sensitive to large errors. MAE and RMSE range from 0 to ∞, and their smaller values indicate better performance of the prediction method.

C. Accuracy Comparison

To demonstrate the prediction accuracy of our method, we reproduce the 6 most representative QoS prediction methods and compare the NBMF with them.

- UMEAN: This method employs the average QoS value of a user to predict the unknown QoS value.
- IMEAN: This method employs the average QoS value of a server to predict the unknown QoS value.
- UIPCC [17]: This method is user-based and service-based CF, which employs similar users and similar services for QoS prediction.
- PMF [18]: This method is probability-based MF, which factorizes the matrix into two low-rank latent feature matrices for QoS prediction.
- HMF [7]: This method improves the MF with location clustering.
- AMF [10]: This method improves the MF with user bias and service bias.
- NBMF: This method improves the MF with network bias.

[2]https://github.com/wsdream/wsdream-dataset

TABLE I
QOS PREDICTION ACCURACY (SMALLER MRE AND RMSE INDICATE BETTER ACCURACY)

Methods	MAE					RMSE				
	D = 15%	D = 20%	D = 25%	D = 30%	Improve	D = 15%	D = 20%	D = 25%	D = 30%	Improve
UMEAN	0.8739	0.8738	0.8737	0.8735	47.95%	1.8569	1.8568	1.8564	1.8576	36.60%
IMEAN	0.6821	0.6805	0.6789	0.6781	33.11%	1.5366	1.5305	1.5280	1.5267	23.08%
UIPCC	0.5861	0.5768	0.5725	0.5714	21.14%	1.4464	1.4322	1.4262	1.4247	17.81%
PMF	0.5236	0.5015	0.4904	0.4652	8.16%	1.2556	1.2303	1.2185	1.1949	3.88%
AMF	0.5024	0.4836	0.4589	0.4444	3.72%	1.244	1.2196	1.1977	1.1772	2.68%
HMF	0.4994	0.4713	0.4545	0.4438	2.67%	1.2354	1.2162	1.1819	1.1582	1.73%
NBMF	**0.4847**	**0.4579**	**0.4459**	**0.4306**	-	**1.2232**	**1.1843**	**1.1627**	**1.1388**	-

In the real world, the user-service matrix is very sparse, as users usually invoke only a small number of Web services. In this paper, to simulate the matrix environment with different densites, we randomly remove a certain number of QoS values from the dataset to generate the user-service matrix with densities of 15%, 20%, 25%, 30%. The removed QoS values are used as expected values to evaluate the prediction accuracy achieved by different methods. For example, a matrix density of 15% means that we randomly select 15% QoS values in the original matrix to predict the remaining 85% QoS values.

In the experiments, the parameters of the baseline methods are initialized according to the corresponding papers for optimal performance, and the parameters of our NBMF method are set to $\alpha = 0.6$, $d = 6$, $\lambda = 0.02$, $\eta = 0.003$, the maximum number of iterations in the model training is set to 300. In addition, we perform an early stopping strategy during the model training, where we stop training if the evaluation metric on the testing set increases five times in a row.

Table I provides the prediction accuracies of different methods at 15% to 30% matrix density. We can observe that NBMF achieves an improvement of 2.67∼47.95% in MAE and 1.73∼36.6% in RMSE compared with other classical prediction methods, and NBMF has the smallest MAE and RMSE regardless of matrix density, which indicates that our method has the best prediction accuracy. Compared with the AMF method, NBMF method achieves 3.72% and 2.68% improvement in MAE and RMSE. Because the real-world network environment is very complex, it is more suitable to consider network bias than user bias and service bias for real QoS prediction systems. Compared with the HMF method, NBMF method achieves 2.67% and 1.73% improvement in MAE and RMSE. Because the HMF method clusters based on regions, while the NBMF method clusters based on network paths, which is more adaptable to complex network environments. The prediction accuracy of all methods improved significantly as the matrix density increased from 15% to 30%, suggesting that more QoS information can contribute to higher prediction accuracy.

D. Impact of Parameter α

The parameter α measures the degree of network bias in our prediction model. If α is set to 0, our prediction model does not consider network bias, in which case NBMF is equivalent to PMF. If α is set to 1, our prediction model does not consider the user-service interaction and only uses linear regression for prediction. To evaluate the impact of α and find an optimal model, we set the dimension d to 6 and set the density D to 15% and 30%.

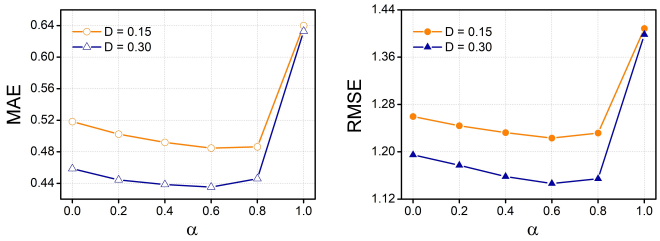


Fig. 2. Impact of Parameter α

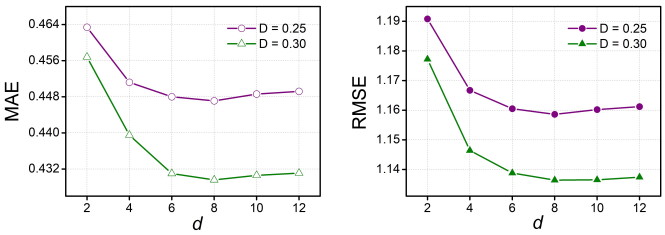


Fig. 3. Impact of Dimension d

Fig. 2 shows us the changes in MAE and RMSE when α is adjusted from 0 to 1. Before the prediction accuracy reaches the best, the values of MAE and RMSE decrease as the value of α increases, indicating that the prediction accuracy is improved. However, when the value of α exceeds a certain threshold, the values of MAE and RMSE increase instead, indicating a decrease in prediction accuracy. We observe that the thresholds for both MAE and RMSE are around $\alpha = 0.6$ for all densities of the matrix environment. The existence of the thresholds confirm our intuition that the best prediction performance can be achieved by a proper combination of MF and network bias. In addition, we find that our NBMF method is quite stable, as it maintains similar trends for different criteria in all configurations.

E. Impact of Dimension d

In our proposed method, d denotes the dimension of the low-rank latent feature matrix, i.e., the number of latent features in matrix factorization. If d is small, only a few key latent features determine the QoS value. If d is large, many latent features jointly determine the QoS value. To investigate the impact of the dimension d on prediction results, we set the parameter α to 0.6 and set the density D to 25% and 30%.

Fig. 3 presents the changes in MAE and RMSE when the dimension d is adjusted from 2 to 12. As the dimension increases, the values of MAE and RMSE decrease rapidly at first, indicating that only a few latent features cannot achieve good prediction results. However, when the dimension exceeds a certain threshold, the values of MAE and RMSE gradually increase. Because higher dimension leads to overfitting problems, which reduces the prediction performance.

V. Related Work

In recent years, collaborative filtering (CF) has been widely used for QoS prediction [8], [9]. Existing CF-based prediction methods can be classified into two types: memory-based CF and model-based CF. Memory-based CF first finds similar users or services by Pearson correlation coefficient (PCC), and then uses the QoS values of similar users or services to predict the missing values [5], [17]. However, a user may have invoked only few services in the real world, which reduces the accuracy of calculating similarity using PCC [1]. Model-based CF trains a global model to make predictions based on observed historical invocation records, and it performs well when dealing with sparse user-service matrix [3].

Matrix factorization (MF) is arguably the most popular model-based CF technique [8], [9], which attempts to capture the interaction between users and services [10], [11]. He et al. [7] introduced a hierarchical MF method based on location grouping, they assumed that local invocations reflect more interaction than global ones. Zhang et al. [2] proposed a neighborhood-integrated MF method, which uses PCC to calculate the similarity between users. Tang et al. [19] employed the IP addresses and Ryu et al. [20] employed the location information to improve the calculation of similar users, they assumed that similar users have similar interaction with services. In addition to the user-service interaction, there are many factors unrelated to the interaction in the real-world prediciton. Although Zhu et al. [10] propose using user bias and service bias to capture the non-interaction from users and services, they do not take into account the non-interaction from the network environment.

VI. Conclusion

We propose a network biased matrix factorization method for QoS prediction. Our method considers both interaction and non-interaction between users and services, which makes our method adaptable to complex network environment. We conduct extensive experiments on a real-world QoS dataset. The result demonstrates that our method can achieve better performance than the state-of-the-art baseline methods.

In the future, we intend to improve the current work as follows: First, we will conduct experiments to evaluate the prediction performance of NBMF on other QoS attributes. Second, considering the dynamic nature of Web services, we will try to implement real-time QoS prediction with weighing factor based on time preference.

References

[1] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng, and Y. Yang, "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, 2019.

[2] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2012.

[3] F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang, "Outlier-resilient web service qos prediction," in *Proceedings of the Web Conference 2021*, 2021, pp. 3099–3110.

[4] J. El Hadad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 73–85, 2010.

[5] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *Ieee international conference on web services (icws 2007)*. IEEE, 2007, pp. 439–446.

[6] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "Qos prediction of web services based on two-phase k-means clustering," in *2015 ieee international conference on web services*. IEEE, 2015, pp. 161–168.

[7] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for web service recommendation," in *2014 IEEE international conference on web services*. IEEE, 2014, pp. 297–304.

[8] S. H. Ghafouri, S. M. Hashemi, and P. C. Hung, "A survey on web service qos prediction methods," *IEEE Transactions on Services Computing*, 2020.

[9] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service qos prediction via collaborative filtering: A survey," *IEEE Transactions on Services Computing*, 2020.

[10] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online qos prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, 2017.

[11] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service qos prediction with location-based regularization," in *2012 IEEE 19th international conference on web services*. IEEE, 2012, pp. 464–471.

[12] A. Klein, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the cloud," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 959–968.

[13] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[14] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, pp. 1–24, 2010.

[15] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[16] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 83–90.

[17] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *2009 IEEE International Conference on Web Services*. IEEE, 2009, pp. 437–444.

[18] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, 2007.

[19] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.

[20] D. Ryu, K. Lee, and J. Baik, "Location-based web service qos prediction via preference propagation to address cold start problem," *IEEE Transactions on Services Computing*, 2018.

A Model Based Approach for Generating Modular Manufacturing Control Systems Software

Mahmoud El Hamlaoui
ENSIAS, Mohammed V University in Rabat
Rabat, Morocco

Yassine Qamsane
Siemens Technology, Automation Engineering Software
Charlotte, NC 28273, USA

Youness Laghouaouta
National Institute of Posts and Telecommunications
Rabat, Morocco

Anant Mishra
Siemens Technology, Automation Engineering Software
Charlotte, NC 28273, USA

***Abstract*—Digitalization is transforming manufacturing systems to become more agile and smart thanks to the integration of sensors and connection technologies that help capture data at all phases of a product's life cycle. Digitalization promises to improve manufacturing flexibility, quality, productivity, and reliability. However, there is still a significant need of effective methods to develop models that could enhance the capabilities of manufacturing systems. Formal methods and tools are becoming essential to achieve this objective. Within this context, this paper introduces a formal software solution for the automatic generation of modular manufacturing control systems software. The proposed solution leverages software Model-Based Design (MBD) techniques to reduce development effort, time, and human error by automating several manual steps.**

Keywords-component; Industry 4.0; Cyber-Physical Manufacturing; Formal methods; MDE; DSL; Grafcet

I. INTRODUCTION

The world is in the midst of a new industrial revolution, referred to as Industry 4.0 or Smart Manufacturing (SM), driven by the rapid advancement in Information Communication Technologies (ICT) in terms of speed, power, autonomy, and mobility. The proliferation of ICT in the manufacturing domain has brought about Cyber-Physical Manufacturing Systems (CPMS), that combine physical production components (e.g., robots, machines, conveyors, and parts to be processed) with cyber components (e.g., logic controllers, networks, and data management infrastructures). CPMS gain their intelligence thanks to their connectivity to the Industrial Internet of Things (IIoT), which is an information network of physical objects (e.g., sensors and machines) that allows interaction and cooperation of these objects to reach common goals [1][2]. CPMS enable more flexibility, reactivity, proactivity, and adaptability to the dynamic global market shifts and fast changing customer requirements.

For manufacturers to remain competitive, CPMS are required to strategically adapt to factory changes associated with the responsiveness to the rapid changes in customer requirements and market conditions over time. In such agile environments, defining the interaction between the manufacturing control system and the physical manufacturing units using conventional development processes, which are characterized by an overhead of manual work, is effort and time-consuming and would have negative impacts on finances. Thus, to address uncertain and emerging situations, the manufacturing control system should be highly flexible, adaptable, and reconfigurable.

In previous work [3][4][5[6], we introduced a formal approach to automatically synthesize modular / distributed supervisory control for CPMS. The approach divides the overall control problem into local and global controls. Local Controllers (LCs) are developed for individual subsystems, then global interactions are added to the LCs to cooperatively execute the overall control actions. This approach ensures the flexibility required in CPMS to adapt to rapid changing conditions. For instance, in case of redesign or system extension, the practitioner would modify only the LCs for the impacted production modules and extend the global specifications with the new requirements, then automatically synthesize the new control logic. In contrast to a centralized controller approach where the entire control system needs to be rebuilt, the proposed approach requires to update only the affected manufacturing units. This approach enables the manufacturing units to be versatile and reusable in different environments.

The contribution of this paper is to derive a systematic software solution that is based on the previously proposed formal approach to support CPMS control designers in automatically generating control systems. The software solution leverages Model-Based Design (MBD) techniques to reduce development effort, time, and human error by automating several manual steps.

The rest of this paper is organized as follows. Section II recalls the theoretical proposition. Section III details the design and development of the proposed software solution. Finally, Section IV summarizes the contributions of this paper and presents some future research avenues for this work.

DOI reference number: 10.18293/SEKE2022-151

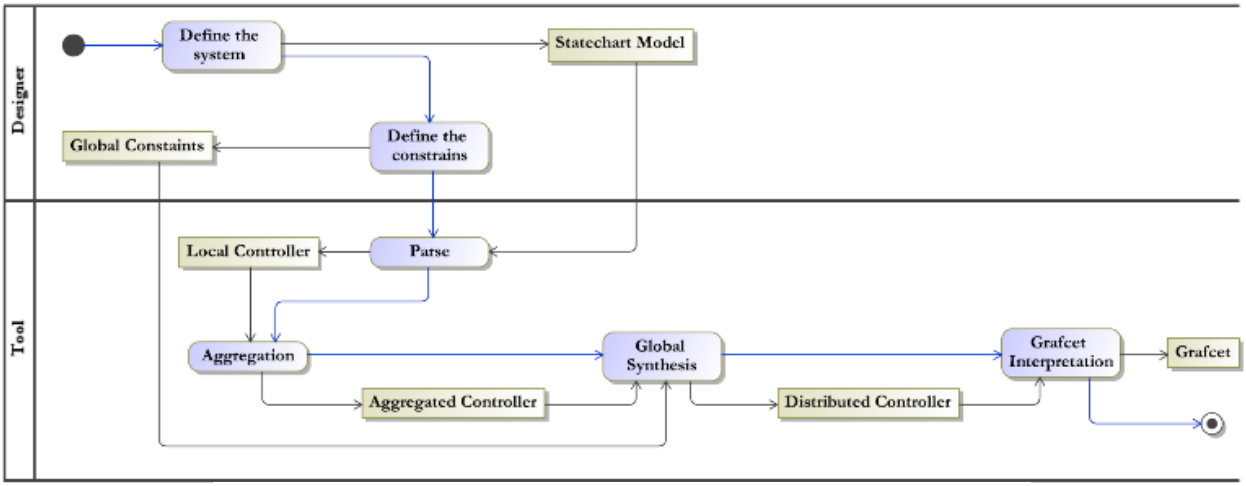


Figure 1. Activity diagram for the synthesis of distributed supervisory control

II. PROPOSED THEORETICAL APPROACH

The workflow shown in Fig. 1 displays the different steps of the supervisory controller synthesis process.

First, the discrete operation physically realizable by the system is modeled using discrete automata models in a local modular way according to its mechanical and functional characteristics (e.g., sensors and actuators). A Local Controller (LC) is derived for each local plant component from its automaton model. Second, to enable the overall system coordination, global specifications over the local components are defined and formalized as logical Boolean expressions. Then, the LC are aggregated to enable applying the logical Boolean expressions to the automata models. The aggregation procedure is carried out such that the states reached by controllable (actuator) events ($\sigma \in \Sigma_c$) are merged into macro-states that are interconnected with uncontrollable (sensor) events ($\sigma \in \Sigma_{uc}$) as explained in Section III. The output of this endogenous transformation is saved as an ALC (Aggregated Local Controller).

The Global synthesis activity consists of applying the formal global specifications to the corresponding ALC models, which allows the local components to coordinate among each other to reach the global control objective. At the output of this activity is the DC (Distributed Controller) models that implement both the local and coordination controls. A single DC is generated for each local manufacturing unit in the system. The last activity transforms the DCs into Grafcet specification in order to exploit the resulting model in the PLC (Programmable Logic Controller). The implemented transformation produces a JSON model visualized as Grafcet into GoJS [7]. The latter is a JavaScript library for creating and manipulating diagrams, charts, and graphs.

III. PROOF OF CONCEPT

According to the activity diagram of Fig. 1, after defining the system and control specification models, the proposed tool consists of four automated steps. First, the designer defines the system using UML Statechart models. Our statechart models are defined using PlantText [8], which is an Open-source tool, based on PlantUML [9], that uses a textual DSL to create graphical UML diagrams. Thus, our proposal is generic and not oriented for a particular software vendor (e.g., MagicDraw, Rational Software Modeler, Modelio, Visual Paradigm, etc.). In order to have a fully integrated MDE approach [10], we have proposed

```
7  Plstatemachine:
8      BEGINFILE (transitions+=Transition)* ENDFILE ;
9
10
11 Transition:
12     NL source= (State | InitialState) SourceToTarget  target=(State | InitialState) SymboleEvent event=ID ;
13
14 State:
15     name=ID ;
16
17 InitialState:
18     name='[*]' ;
19
20 terminal NL: ('\n')+;
21 terminal BEGINFILE:'@startuml';
22 terminal ENDFILE:NL'@enduml';
23
24 terminal SourceToTarget: '-->';
25 terminal SymboleEvent : ':';
```

Figure 2. Proposed grammar for PlantText

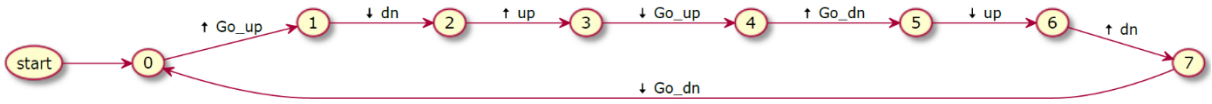


Figure 3. Local Controller (LC) model example

an alternative to the online PlantText editor. For this, we propose a textual editor based on Xtext. Figure. 2 shows the implemented grammar of the textual editor. Second, the aim of the "Parse" activity is to take the UML Statechart model established by the designer and automatically produce a LC model (Fig. 3) that is conform to a General Controller Metamodel.

The transformation is expressed as graph transformation unit implemented using Henshin [11]. Henshin is a transformation language and tool environment based on graph transformation concepts and operating on EMF models [11]. It provides features needed to express complex transformation such as negative application conditions (NACs), which specify the non-existence of model patterns in certain contexts and transformation units to control the rules application sequence. We have to notice that the rule declaration in the Henshin formalism does not explicit the description of the left- and right-hand sides. Instead, it is based on the following stereotypes to depict the rule application semantic: preserve, create, require, and forbid

Third, the "Aggregation" activity is an endogenous transformation that takes as input the LC model and produces the corresponding ALC model. The aim of this transformation is to apply corresponding control specifications to the latter. The ALC is produced through a graph transformation implemented with Henshin. Figure. 4 shows the rules being used to aggregate states respectively according to *Inh* and *Ord*. Transformations consist of removing the controllable evolutions from the LC model, and joining them into macro-states as follows: if the controllable event is associated with a rising edge, then the order is authorized and belongs to the set $Ord^{(DC)}$; otherwise, the order is inhibited and belongs to the set $Inh^{(DC)}$. Figure. 5 shows the resulted ALC model.

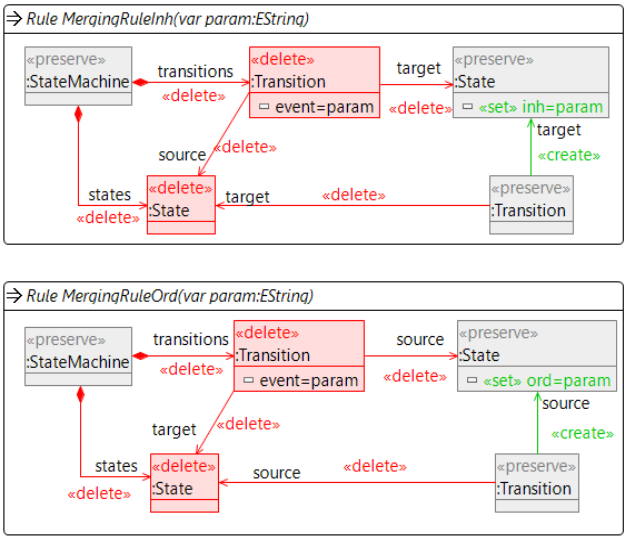


Figure 4. Aggregation transformation rule in Henshin

The basic data structure of a global constraint is a Boolean expression of the following form:

If (Condition) **Then** (Action)

Formally, the set of global constraints is defined by the pair $Spec = (C^{(spec)}, Act^{(spec)})$, where $C^{(spec)}$ is the set of *conditions*; and $\text{Act}^{(\text{spec})} = \{\text{Ord}^{(\text{spec})}, \text{Inh}^{(\text{spec})}\}$ is the set of activation/deactivation *actions*. Due to space limit, we choose to not present the textual editor grammar as it is based on the same idea of Fig. 2.

Example global constraints to be applied to the LC of Fig. 3 are stipulated as presented in Fig. 6. The global constraints are added to the ALC as follows. Check all the constraints for each ALC state. If an authorized (resp., inhibited) order of an ALC state is similar to that authorized (resp., inhibited) within a global constraint, then the constraint's condition should be associated with the actual state to condition the authorization (resp., the inhibition) of the corresponding order. The resulting controller is a DC model (conforms to the General Controller Metamodel). To avoid any confusion between the LC and ALC metamodels, the values regarding the conditions *Ord_If* and *Inh_If* are only represented in the DC model.

The rules implied in the transformation unit that takes as input the ALC and constraint models and produces the DC model as output (Fig. 7) have been implemented using Henshin. Dedicated rules have been expressed in order to retrieve *Inh* and *Ord* nodes from the constraint models and set the adequate values of *Inh_If* and *Ord_If* attributes.

The purpose of the last activity is to create a Grafcet model to be used in a running system (the PLC). To do so, we have used the metamodel proposed in [12]. It defines the steps and the different transitions between them.

To avoid defining a concrete representation for the Grafcet model, we decide to visualize the output into GoJS [5]. For that, the transformation is defined to a JSON model. For the sake of brevity, we don't explicit the transformation rules in this paper. Figure. 8 presents the visual model being generated for the LC of Fig. 3.

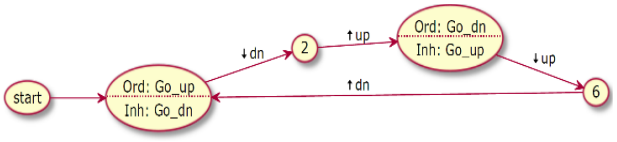


Figure 5. Aggregated Local Controller ALC model example

Id	If	Then	
	$C^{(spec)}$	$Ord^{(spec)}$	$Inh^{(spec)}$
1.	$free \wedge \neg Eject \wedge wpa2$	Go_up	
2.	$(\neg Ok \wedge \neg Eject) \vee ((Ok \wedge Eject) \rightarrow \uparrow ebp)$	Go_dn	

Figure 6. Aggregation transformation rule

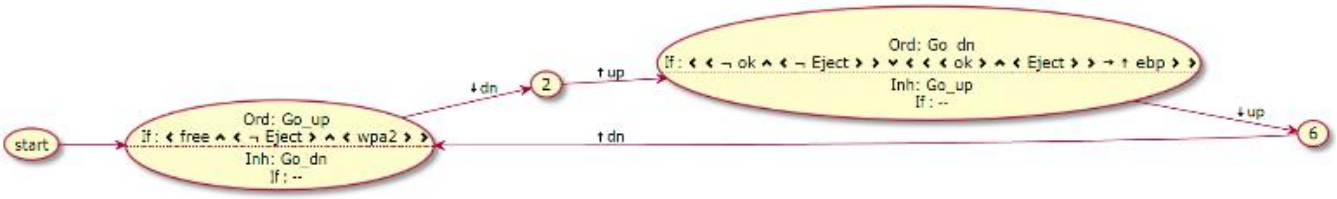


Figure 7. Resulting DC model example

IV. CONCLUSION & PERSPECTIVES

This paper presents a tool for automating the process of generating distributed supervisory control interpreted as Grafcet specification (IEC 60848 [13]) for CPMS. The tool framework decreases the state space explosion problem inherent to formal supervisory control theory methods by using a modular/distributed structure that avoids the synchronous composition of subsystem models. It also increases the flexibility required in manufacturing systems through distributed control models that enable a simple and adaptive control strategy, i.e., in the case of a redesign, only a small amount of data related to the corresponding subsystem controllers will be updated.

The founding framework of the tool uses model-checking technique for the verification of absence of deadlocks and making sure that the system safety and liveness requirements are met before the Grafcet implementation of the distributed control models. Model-checking technique allows tracing the sequences altering the system behavior, which enables easy update of the distributed control models. After the verification phase, simulation is used to validate the distributed control behavior.

We believe that additional investigations of the verification/validation process are of significant importance and would bring improvements to our software tool.

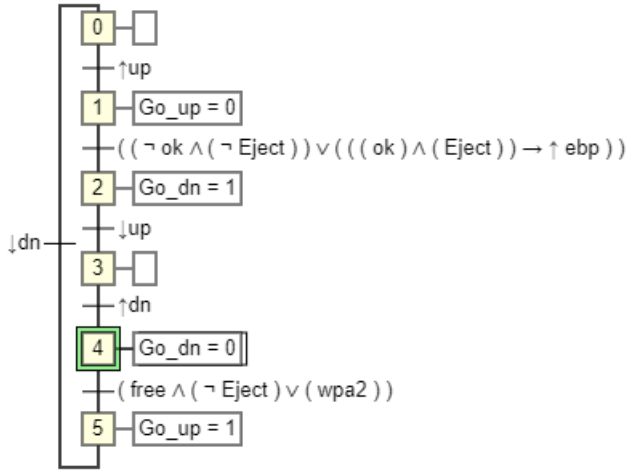


Figure 8. The generated Grafcet

Extensions of this work would introduce a linkage between our software tool and model-checking technique. Another avenue of research is the introduction of time-domain to the framework of this paper to evaluate quantitative control problems and measure system performance.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," IEEE transactions on industrial informatics, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] Y. Qamsane, E. C. Balta, J. Moyne, D. Tilbury, and K. Barton, "Dynamic rerouting of cyber-physical production systems in response to disruptions based on sdc framework," in 2019 American Control Conference (ACC). IEEE, 2019, pp. 3650–3657.

[3] Y. Qamsane, A. Tajer, and A. Philippot, "A synthesis approach to distributed supervisory control design for manufacturing systems with grafcet implementation," International Journal of Production Research, vol. 55, no. 15, pp. 4283–4303, 2017.

[4] Y. Qamsane, M. E. Hamlaoui, A. Tajer, and A. Philippot, "A tool support to distributed control synthesis and grafcet implementation for discrete event manufacturing systems," IFAC-PapersOnLine, vol. 50, no. 1, pp. 5806–5811, 2017.

[5] Y. Qamsane, M. E. Hamlaoui, A. Tajer, and A. Philippot. "A model-based transformation method to design PLC-based control of discrete automated manufacturing systems." 4th International Conference on Automation, Control Engineering and Computer Science (ACECS-2017). Vol. 19. 2017.

[6] Y. Qamsane, A. Tajer, and A. Philippot. "Towards an approach of synthesis, validation and implementation of distributed control for AMS by using events ordering relations." International Journal of Production Research 55.21 (2017): 6235-6253.

[7] N. Software, "Gojs, a javascript library for html diagrams," 2019.

[8] P. U. editor, "Uml editor - an online tool that generates images from text," https://www.planttext.com/, 2008.

[9] A. Roques, "Plantuml: Open-source tool that uses simple textual descriptions to draw uml diagrams," 2015.

[10] M. E. Hamlaoui, S. Bennani, M. Nassar, S. Ebersold, and B. Coulette, "Heterogeneous design models alignment: from matching to consistency management," in 33rd ACM/SIGAPP Symposium On Applied Computing (SAC 2018). ACM Digital Library, 2018, pp. 1695–1697.

[11] T. Arendt, E. Biermann, S. Jurack, C. Krause, and G. Taentzer, "Henshin: advanced concepts and tools for in-place emf model transformations," in International Conference on Model Driven Engineering Languages and Systems. Springer, 2010, pp. 121–135.

[12] P. Guyard, "Atl transformation example : Bridging grafcet, petri net, pnml and xml," 2005.

[13] IEC-60848, "Grafcet specification language for sequential function charts," 2013.

Revealing Agile Mindset Using LEGO® SERIOUS PLAY®: Experience from an Online Agile Training Project

Ilenia Fronza Xiaofeng Wang
Free University of Bozen/Bolzano, Italy
{ilenia.fronza, xiaofeng.wang}@unibz.it

Abstract

LEGO® SERIOUS PLAY® (LSP) is an effective methodology to enable the representation of abstract concepts and has been applied to teach several Software Engineering topics. However, there is limited evidence on how LSP can be used in training on agile mindset, which is the core and central element of agile methods. This paper demonstrates how LSP can be utilized in agile training to reveal the agile mindset of participants. We describe our experience of utilising LSP in an agile training project for local software companies. Since the project was run during the COVID-19 pandemic, we adapted the LSP methodology for online settings, which was not straightforward because face-to-face interactions and tangible objects are key characteristics of LSP. In this experience report, we describe the design of the online LSP workshops and explain how to analyze LSP models to reveal the agile mindset of the participants and to tailor the training accordingly. We also provide evidence of the effectiveness of the LSP methodology in our training project. Drawing upon our experience, we synthesize a set of lessons learned and sketch recommendations for educators who intend to apply LSP in their future endeavours.

1 Introduction

LEGO bricks are usually associated with constructing concrete structures such as buildings or cars, but the dynamic nature of LEGO also allows it to represent more abstract concepts: LEGO® SERIOUS PLAY® (LSP) is a facilitated workshop where participants build three-dimensional models using a special mix of LEGO bricks designed to inspire the use of metaphors and story-making [1].

One specific Software Engineering (SE) subject that is frequently taught using LEGO is agile methods (e.g., [2, 3]). It is commonly understood that agile methods are not just about processes, practices, and tools but, more crucially, an agile mindset, including customer focus, iterative and incremental way of working, fast feedback loop, and continuous learning and improvement [4]. Due to the abstract nature of the agile mindset, it is not surprising to observe that there are limited studies on explicit training for it. The potential of LSP working with abstract concepts could be exploited for this purpose. The objective of this experience report is to demonstrate how to utilize the LSP methodology in agile training that has the agile mindset as a core element, especially how it helps to reveal the participants' agile mindset. We describe our experience of utilizing LSP in an agile training project we run for local software companies. Since the training project has been carried out during the COVID-19 pandemic, we adapted the standard LSP methodology for online settings. It rendered as a challenge because face-to-face (F2F) interactions are key characteristics of the LSP methodology, but limited alternative solutions have been proposed. Thus, our experience report also shows a feasible implementation of LSP in online settings.

The rest of the paper is organised as follows. Section 2 provides more details on LSP and defines the agile mindset. Section 3 is the core of this experience report. After outlining the agile training project as the context, it elaborates on the configuration and online adaptation of the two LSP workshops and explains how to analyze the resulting LSP models to reveal the agile mindset of the participants and to tailor the training. The evidence on the effectiveness of the LSP methodology in our training project is also provided in the same section. In Section 4, we synthesize a set of lessons learned and sketch recommendations for educators who intend to apply LSP in their future endeavours. Future work is outlined in the conclusion section.

2 Background and related work

The idea behind LSP is that building external models that can be examined, shared, and discussed makes it easier to construct internal mental maps [1].Other key theories behind LSP are the importance of play as a way to learn [5] through exploration and storytelling; the hand-mind connection as a new path for creative and expressive thinking; and the role of the different kinds of imagination [6]. To

DOI reference number: 10.18293/SEKE2022-117

be considered as such, LSP workshops must follow the LSP Core Process, which is based on four essential steps: 1) The facilitator poses a challenge having no obvious/correct solution (e.g., "What's your worst nightmare for this design outsourcing initiative?" [1]); 2) Participants build their answers by assigning a meaning to bricks and develop a story covering the meaning; 3) Participants share their stories; and 4) Participants reflect on what has been shared. A workshop typically takes from half a day to a couple of days. The first part serves to familiarize with the core process; then, each workshop combines a selection of seven techniques (see [6] for details). LSP efficacy has been observed in maintaining student energy, engagement, and concentration [7]. Hence, LSP has been successfully applied in multiple education areas [8], such as information systems management [9], creative arts [10], HCI design [11], civil engineering [12], and industrial engineering [13].

LSP in SE training and education. According to a separate systematic literature analysis we conducted (details are given in Appendix), many studies used LEGO bricks in SE training but only six papers used the LSP core process. The small number seems to contrast with the great success of LSP: more than 10,000 LSP facilitators [14] write white papers/blogs, and dedicated events exist. LPS has been applied to teach the following SE topics: constructing organizational identity [15], requirements engineering and dimension and dependability [8, 16], cross-domain stakeholder-alignment [17], development of a shared vision of the product [18], and team building [19]. However, there is limited evidence on how LSP can be used in training on agile mindset. Moreover, none of the papers proposed online LSP workshops; during the pandemic, limited anecdotes were reported in blogs (e.g., [20]) and validation experiments were suspended (e.g., [17]).

Training on agile mindset. Agile mindset is considered at the core of agile methods and their successful applications, so crucial to the point that Denning (2016) claims: "Agile is primarily a mindset" [21]. It affects all organisational levels and thus needs to be aligned across the whole organisation [21]. Despite of its importance, there is a lack of shared understanding of what agile mindset is. It is an ambiguous term and prone to misinterpretation [4]. Several recent studies aspired to provide a common definition of agile mindset. An opinion survey of 52 agile practitioners [22] evaluated 70 unique agile mindset elements of an effective team. The evaluation results in the top 5 evaluated agile mindset elements for effective teamwork, including searching for a solution to the problem instead of finding the guilty, being motivated, helping each other, mutual listening, and focus on achieving common goal. Based on a systematic review of scientific and grey literature, and semi-structured and unstructured interviews with agile practitioners, Mordi and Schoop [4] consolidate 192 agile mindset elements into 27 final characteristics, and create a definition of agile mindset that comprises the following elements: (AM1) Trust; (AM2) Responsibility and ownership; (AM3) Continuous improvement; (AM4) Willingness to learn; (AM5) Openness and willingness to continually adapt and grow; (AM6) Specific personal attributes, including intent, integrity, honesty, transparency, courage, authenticity, empathy, proactivity, creativity and problem-orientation; (AM7) Enabling environment; (AM8) Autonomy of people and teams; (AM9) Managing uncertainty; and (AM10) Focus on customer value.

The reviewed studies demonstrate that agile mindset is a broad concept and the constituting and interwoven elements cover personality traits, teams, culture, environments, leadership and management, which are all crucial for understanding agile mindset appropriately. Together with the invisible and intangible nature of mindset, it is difficult to provide effective training on agile mindset. This is evidenced by the abundant literature on training either students or professionals for specific agile methods and practices but significantly less literature on training for agile mindset. Many agile training programs have agile mindset training as an implicit rather than an explicit element. For example, Hof et al. [23] use a gamification approach in a multi-week scrum simulation project in an undergraduate software engineering course. Agile values and collaboration are taught to the students implicitly through playing the Scrum Paper City simulation. Often, various agile mindset elements may be considered soft skills and covered by the programs that focus on training professionals on soft skills (e.g., [24]). The power of agile mindset is somehow downplayed when its elements are treated as soft skills, as mindsets are responsible for our behavioral, physiological and psychological responses [25]. LSP provides a unique opportunity for more explicit and holistic treatment of agile mindset in a training program. It inspires workshop participants to create metaphors and make stories with three-dimensional LEGO models that could surface their agile mindsets, which could then help trainers to understand where to focus on for an effective training. As far as the authors are aware of, there are no previous studies that investigated how LSP can be utilized in agile mindset training, let alone in online settings. Our report can be seen as one of the first attempts in this direction, drawing upon our own experiece in offering online training related to agile mindset.

3 Revealing Agile Mindset Using LSP

In this section, first, we outline the training project from which we draw the experience report. Against this backdrop, we describe the configuration and implementation of the two online LSP workshops and our analysis of the LSP models in terms of agile mindset. Finally, we demonstrate the effectiveness of the LSP methodology.

3.1 The Online Agile Training Project

The project provided agile training to 31 participants (28 M, 3 F) from four software companies having the following missions: solutions for the integration of IT system and business process (2 participants); solutions to support everyday digital life and complex business processes (7); web solutions for spatial data management (2); business intelligence, ERP, CRM, cyber security solutions (20). According to our survey (26 respondents), most of the participants (38.5 %) had 10-15 years of working experience, 7.7% 1-5 years, 19.2% 5-10 years, 34.6% more than 15 years. 28.0% never used agile methods, and 12.0% used them for less than one year. Most of them (48%) used agile methods for 1-5 years, mainly at the team level (60.9%).

The project participants were divided into four groups based on their roles in the companies: project managers (8, 2 F), program and middle managers (6), software engineers (14, 1 F), and other roles (e.g., marketing, CEO, human resources) (2). The central part of the training included different topics for each group (e.g., *Advanced SE Techniques* for software engineers), while beginning and ending segments were the same for all the groups. The beginning segment focused on fundamental agile concepts, principles, and practices. The goal was to provide the participants with a common knowledge foundation that could be applied across different contexts regardless of their roles and responsibilities. Understanding fundamental agile values and principles was also crucial for the participants to take part in the final part of the training (i.e., *scaling agile/agile transformation*), which allowed them to understand the agile/lean approach from a broader organizational perspective, and understand the challenges and success factors to scale agile methods to the whole organization. A retrospective session followed the training: all participants of the same company (who were previously in different groups) confronted what they learned and consolidated the acquired knowledge.

3.2 Online LSP Workshop Implementation

We conducted two four-hour LSP workshops, WS1 in the first part (during the *agile fundamentals and mindset* session) and WS2 at the end of the training, seven months later. The detailed description of how the agile mindset was taught is out of scope in this article, which focuses on how we used the core LSP process to collect data on agile mindset (Figure 1). Each LSP workshop was repeated for the four groups; in total, participants were 25 in WS1 and 22 in WS2. The time used for building, sharing, and discussing respected the indications of the LSP methodology and, as expected, varied by group depending, for example, on familiarity with LEGO bricks and the length of discussions.

The facilitator (one of the authors is a certified facilitator for LSP) faced the main challenge of recreating the LSP mood and learning experience (based on "thinking in 3D" [10]) in the online environment (using Zoom as video conference software). We encouraged and motivated [26] camera and mic constant usage to facilitate communication and to hear the "sound of bricks". To guarantee the same learning experience, we distributed the same official LEGO *LSP starter kit* to each participant.

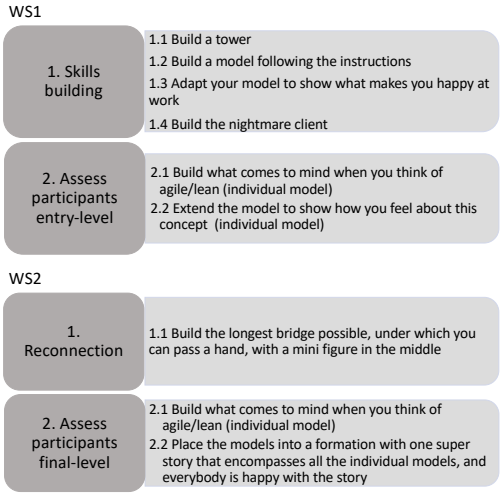


Figure 1: Goals and activities of WS1 and WS2.

Group work is a key component of F2F LSP, for example, for building shared models. In the online workshops, we created super stories that encompassed the individual models, which could be achieved by using the available bricks and did not require having one single builder. The participants uploaded to a shared document the pictures of individual models. Then, to create the super story, they organized the pictures in the document. During story sharing, the facilitator annotated each model in a visual booklet to record how the participants interpreted the elements of their LSP models, and observed the participants' reactions and perceptions. These notes, together with the LSP models, are the input for the analysis of the agile mindset of the participants, which is explained in the following sub-section.

3.3 Utilizing LSP Models to Reveal Agile Mindset

The authors reviewed all the collected materials to surface the agile mindset of the participants by mapping the LSP models and their interpretations to the Agile Mindset (AM) elements. Table 1 shows the mapping of the four examples shown in Figure 2. The story of Model 1 was "agile means reaching the goal by taking decisions to adjust the sails to the wind", which shows *Willingness to adapt* (AM5) and *Autonomy of people and team* (AM8).

To obtain an overall picture of the participants' AM, we thought to create a dashboard to supervise the learning process of a group of learners [27] through an effective and intuitive means [28] to visually display levels of presence

Table 1: Models in Figure 2 mapped to agile mindset elements.

	Agile mindset elements									
Model	1	2	3	4	5	6	7	8	9	10
1					x			x		
2					x			x	x	x
3				x	x			x	x	x
4										x

Figure 2: LSP models created by the participants to answer the question "what is agile?".

of different AM elements revealed by the LSP models. As a first step, we used a heatmap for visualization. Figure 3 exemplifies our approach. Lines and columns represent the two LSP workshops and the AM elements, respectively. Each square represents the percentage of participants who mentioned an AM element in the LSP model, with larger values represented by darker squares. This way, the two lines of the heatmap reveal the AM in the group of participants of the two LSP workshops.

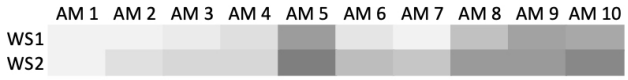


Figure 3: Agile mindset revealed by the LSP workshops.

The proposed visualization would help the trainer to focus on the training on the critical agile mindset elements. A closer examination of the heatmap of LSP WS1 helped us to understand where the presence of AM elements is weaker or even not present, so that we could tailor our training plan to focus more on those elements. For example, as shown in Figure 3, in comparison to other agile mindset elements in the WS1 line, *Trust* (AM1), *Responsibility and ownership* (AM2), and *Enabling environment* (AM7) are least shown in the LSP models from the first workshop. This indicated that we should focus more on these elements in our training. Instead, relatively less attention could be paid to *Openness and willingness to continually adapt and grow* (AM5), *Autonomy of people and teams* (AM8), *Managing uncertainty* (AM9) and *Focus on customer value* (AM10). In addition, the comparison of the two heatmap lines in Figure 3 helped us to understand whether the participants' AM was enhanced through the training project and which elements were enhanced. Most squares in the WS2 line are darker than the counterparts in the WS1 line, which shows stronger presence of those AM elements in the models built in the second LSP workshop. We interpret this as the enhancement of the AM elements. It is reassuring for us to observe that no element in the WS2 line shows reduced presence, which can be interpreted that our training regarding agile mindset did not produce any counter effect, even though we could also see that two elements, *Trust* (AM1) and *Willingness to learn* (AM4), did not show improvement.

3.4 The Effectiveness of LSP to Review Agile Mindset

The LSP models allowed us to compare the participants' AM in the beginning and at the end of the training project. Our experience is that they are more effective in revealing AM than traditional text input. During the project kick-off meeting, we divided all project participants into pairs, and asked them to write down their answer to the question "*what agile means to you?*" and to discuss their answers with their pairs. This provided us a good opportunity to compare the expressiveness of text versus LSP models in terms of revealing the participants' AM. Indeed, most of these participants participated in LSP WS1, and there was no training session in between the kick-off meeting and the first LSP workshop; thus, the AM of the participants were supposedly not changed, and there is sufficient time lapse between these two sessions so any potential influence of pair discussion in the kick-off meeting on the LSP model building in WS1 was minimised. We analysed the text input against the ten AM elements, in the same manner as we annotated the LSP models. A heatmap similar to Figure 3 was generated that compares the revealed AM items between text input and the models from WS1, as shown in Figure 4.

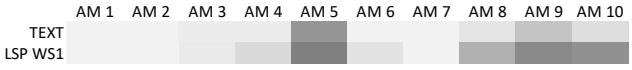


Figure 4: Agile mindset revealed by text input and by the models from LSP WS1.

For the same group of participants, the LSP models produced by them revealed stronger presence of most AM elements (except AM1, AM2 and AM7), especially for AM8, AM9 and AM10 where sharper contrasts can be observed. Figure 4 is a good illustration that LSP models can help educators better surface the invisible AM of the participants.

It is also worth noting the intangible results of the LSP workshops. Based on facilitator's notes on participants' reactions and perceptions, the LSP method helped to keep the

participants engaged, which was an issue in the online setting, and the participants of both workshops were eager to build their models. Of course, some participants were less enthusiastic than others; we did not push them hard, and left them enough freedom and ease to build their own models as they wished. During the workshops, an open atmosphere was created and the LSP methodology helped the team building task. Our experience confirmed what was observed in [18]: even though team building was not the primary goal of the LSP workshops, the *camaraderie* effect was pronounced. Finally, and perhaps most crucially, the participants had fun.

4 Lessons Learned and Recommendations

Drawing upon the experience reported in Section 3, we summarize a set of lessons learned and corresponding recommendations of utilizing the LSP methodology, especially in online settings. We hope they could be useful for other educators for using LSP in.

Running LSP workshops.

- LSP workshops require considerably more time than traditional text-based surveys; however, LSP can better surface participants' invisible agile mindset.
- LSP facilitators need to be aware of the difficulties that the audience might have with building. For example, we had a color-blind participant who had difficulty following the building instructions in WS1. We explained to the participant that instructions were needed only in that specific activity and allowed as much time as needed to complete the construction. We recommend collecting the necessary information about the participants to plan the workshop carefully.
- We recommend gathering participants' feedback to collect tips for replication elsewhere [16].

Running LSP workshops in online settings.

- To ease the workshop execution, we recommend providing each participant with the same LEGO set (as in F2F workshops). However, we recognize that this may be difficult in the case of global participant groups.
- We recommend asking participants to keep cameras and mics on to obtain spontaneous interactions and more realistic environments. We could observe that the *show-and-tell* nature of the exercises with LEGO bricks facilitated this request.
- The time needed for each online activity depends on several factors (e.g., familiarity with LEGO bricks and the length of the discussions). We recommend having backup activities to keep the fastest builders engaged as, in our case, they tended to disengage quickly (e.g., switching windows to read emails).
- We recommend keeping the 12 participants limit per facilitator as in F2F workshops: if the group is larger the LSP Core Process takes too long and it is hard to keep everyone in flow.

Running LSP workshops when training for agile mindset.

- LSP methodology is effective in training for abstract and complex concepts such as the agile mindset. It is helpful to have a concrete definition of the abstract concept, and build a mapping between LEGO pieces and the components of that concept before running LSP workshops. This could greatly help the educators to better grasp the participants' understanding of the concept.
- We demonstrated how to surface the presence of an agile mindset in a participant group. To enable more *personalized* training, we recommend to keep the mapping between participants and the LSP models they build in different workshop sessions in order to compare their initial and final mindset.

5 Conclusion

LSP is an effective methodology to enable the representation of abstract concepts. In this experience report, we described our adaptation of the LSP methodology for an online agile training with local software companies during the COVID-19 pandemic time. We focused on how the LSP methodology could help to reveal the agile mindset of the participants. We also explained how the models produced in LSP workshops could be analysed to provide more focused and targeted training to participants and better understanding of the training effect. The approach presented in our experience report can help educators in agile training for organizations, because it shows a concrete means to understand the presence and level of agile mindset in the organization. Drawing upon our experience, we summarized a set of Lessons learned and recommendations that could help other educators utilize the LSP methodology in a meaningful manner in their training.

Through our experience, we demonstrated the effectiveness of LSP models to reveal the agile mindset of people. However, to establish the effectiveness of the LSP methodology in training for agile mindset, more rigorously designed studies are needed. For example, as in the papers [8, 16], controlled experiments can be used to compare the learning outcome of one group (treatment group) using the LSP methodology and another group (control group) using conventional approaches such as pair/group discussion or questionnaires. Potential interesting aspects to explore include the usage of *show-and-tell* activities (e.g., LSP) as a potential solution to let people keep camera on during

online sessions, how LSP works for different backgrounds (disciplinary, gender, etc.), which would be important because agile mindset is relevant to all the roles in an organization, and whether/how LSP could be used in training for other types of mindsets or abstract concepts as well as concrete skills or knowledge. A follow-up study could examine whether the participants implemented the studied principles in real-life. Finally, the proposed approach could be compared with other approaches to judge the extent to which agile mindset has been developed.

Appendix

In the systematic mapping study we conducted, we retrieved existing works on *LSP in software engineering training and education* in November 2021 by using the following search strings in all metadata: *(lego OR brick OR LSP OR "lego serious play") AND ("software engineering" OR agile))*. We found 186 works from three digital libraries: IEEE Xplore (126), ACM DL (25) and Scopus (35). After duplicates removal, based on title/abstract we excluded: 1) Studies dealing with programming/robotics; 2) Studies with no focus on SE education; 3) Studies not presented in English; 4) Summaries of conferences/editorials; 5) Studies not accessible in full-text; 6) Replications; 7) Books. We excluded one of the six remaining papers after full-text reading because it did not use the LSP core process. One paper was then added through backward snowballing [29].

References

[1] R. Rasmussen, "When you build in the world, you build in your mind," *Design Management Review*, vol. 17, no. 3, pp. 56–63, 2006.

[2] M. Paasivaara, V. Heikkilä, C. Lassenius, and T. Toivola, "Teaching students scrum using lego blocks." ACM, 2014, p. 382–391.

[3] J.-P. Steghöfer, E. Knauss, E. Alégroth, I. Hammouda, H. Burden, and M. Ericsson, "Teaching agile - addressing the conflict between project delivery and application of agile methods," in *2016 IEEE/ACM 38th Int. Conf. on Software Engineering Companion (ICSE-C)*, 2016, pp. 303–312.

[4] M. Mordi, Azuka; Schoop, "Making It Tangible - Creating a Definition of Agile Mindset," *Twenty-Eigth European Conference on Information Systems (ECIS2020)*, no. June, 2020.

[5] I. Fronza, N. E. Ioini, and L. Corral, "Teaching computational thinking using agile software engineering methods: A framework for middle schools," *ACM Transactions on Computing Education (TOCE)*, vol. 17, no. 4, pp. 1–28, 2017.

[6] E. Frick, S. Tardini, and L. Cantoni, "White paper on lego serious play: A state of the art of its applications in europe, 1–26," *Switzerland: Università della Svizzera Italiana*, 2013.

[7] D. López-Fernández, A. Gordillo, F. Ortega, A. Yagüe, and E. Tovar, "Lego©serious play in software engineering education," *IEEE Access*, vol. 9, pp. 103 120–103 131, 2021.

[8] S. Kurkovsky, "Teaching software engineering with lego serious play," in *Proc. of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, p. 213–218.

[9] B. Oberer, "Integrating creative learning elements in higher education shown in the example of a management information systems courses," *Education*, vol. 3, no. 6, pp. 319–324, 2013.

[10] A. R. James, "Lego serious play: a three-dimensional approach to learning development," *J. of Learning Development in Higher Education*, no. 6, 2013.

[11] L. Cantoni, L. Botturi, M. Faré, and D. Bolchini, "Playful holistic support to hci requirements using lego bricks," in *International Conference on Human Centered Design*. Springer, 2009, pp. 844–853.

[12] L. Bulmer, "The use of lego® serious play in the engineering design classroom," *Proc. of the Canadian Engineering Education Association (CEEA)*, 2009.

[13] P. K. Hansen and R. O'Connor, "Innovation and learning facilitated by play," *Encyc. of the Sciences of Learning*, pp. 1569–1570, 2012.

[14] D. Lyons, "Why Have Our Offices Become Like Touchy-Feely Kindergartens?" https://forge.medium.com/, 2018, accessed: Mar. 2022.

[15] D. David and D. J. Roos, "Constructing organizational identity," *Academy of Management Annual Meeting, NewOrleans, US*, 2004.

[16] S. Kurkovsky, S. Ludi, and L. Clark, "Active learning with lego for software requirements," in *Proc. of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: ACM, 2019, p. 218–224.

[17] J. Köhlke, S. Hanna, and J. Schütz, "Cross-domain stakeholder-alignment in collaborative sos – lego©serious play©as a boundary object," in *2021 16th International Conference of System of Systems Engineering (SoSE)*, 2021, pp. 108–113.

[18] D. Pichlis, S. Hofemann, M. Raatikainen, J. Sorvettula, and C. Stenholm, "Empower a team's product vision with lego©serious play©," *Lecture Notes in Computer Science*, vol. 9459, pp. 210–216, 2015.

[19] B. A. Scharlau, "Games for teaching software development," in *Proc. of the 18th ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2013, p. 303–308.

[20] E. Conches, "Case study lego©serious play©shared model online," 2020, last accessed: Mar. 2022.

[21] S. Denning, "How to make the whole organization "Agile"," *Strategy and Leadership*, vol. 44, no. 4, pp. 10–17, jan 2016.

[22] J. Miler and P. Gaida, "On the agile mindset of an effective team - An industrial opinion survey," *Porc. of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019*, vol. 18, no. 2, pp. 841–849, 2019.

[23] S. Hof, M. Kropp, and M. Landolt, "Use of gamification to teach agile values and collaboration: A multi-week scrum simulation project in an undergraduate software engineering course," in *Proc. of the 2017 ACM Conf. on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2017, p. 323–328.

[24] P. Kontodiakou and A. Sotiropoulou, "Training of ict professionals in soft skills: The case of sending," in *24th Pan-Hellenic Conference on Informatics*, ser. PCI 2020. ACM, 2020, p. 354–358.

[25] A. J. Crum, K. A. Leibowitz, and A. Verghese, "Making mindset matter," *BMJ (Online)*, vol. 356, no. February, pp. 1–4, 2017.

[26] F. R. Castelli and M. A. Sarvary, "Why students do not turn on their video cameras during online classes and an equitable and inclusive plan to encourage them to do so," *Ecology and Evolution*, vol. 11, no. 8, pp. 3565–3576, 2021.

[27] I. Fronza and C. Pahl, "Envisioning a computational thinking assessment tool," in *CEUR Workshop Proceeding*, vol. 2190, 2018.

[28] S. Few, *Information Dashboard Design: Displaying data for at-a-glance monitoring*. Analytics Press Burlingame, CA, 2013, vol. 5.

[29] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

Quantum Software Models: Software Density Matrix is a Perfect Direct Sum of Module Matrices

Iaakov Exman and Alexey Nechaev
Software Engineering
The Jerusalem College of Engineering, JCE, Azrieli
Jerusalem, Israel
iaakov@jce.ac.il, alosha82@gmail.com

***Abstract*— *Quantum Software Models* is a theoretical framework to systematically design and analyze any software system – be it quantum, classical or hybrid – representing it by a design Density Matrix. Recently, we have demonstrated a top-down approach, to decompose a whole software system Density Matrix into modules, using basis vector projectors of the Matrix. However, it would be even more natural to have a systematic bottom-up procedure, to compose a whole software system Density Matrix, given a set of well-designed software module matrices, taken as sub-systems. This is exactly the paper's purpose. The result obtained: the whole software system Density Matrix is a perfect *Direct Sum* of module density matrices. This result yields clear software design benefits: it is bidirectional, one can traverse the software system hierarchy top-down or bottom-up, in particular, gradually building up the whole system from verified correct modules, assured by spectral decoupling techniques. The claim is formally validated and is illustrated by software system studies.**

Keywords*— *Software Design; Software Density Matrix; Modularity; Direct Sum; Module Density Matrix; Quantum Software.

I. INTRODUCTION

Quantum Software Models is a theory of software system design, consisting of a surprising synthesis of two apparently unrelated knowledge frames, both based upon linear algebra.

The **1st** knowledge frame, *Linear Software Models* [6], [7], a linear algebraic software design theory, where vectors stand for software concepts, combined into software system matrices, such as the Laplacian [24]. Matrices enable software ***systems decomposing into modules*** by spectral methods [8].

The unexpected synthesis starts with Frederick Brooks' original idea: "*Conceptual Integrity* is the most important consideration for software system design" [4]. *Linear Software Models* express Brooks' underlying conceptual principles in algebra: ***Propriety*** – use only absolutely necessary concepts to describe software and no more – implying vectors' *linear independence*. ***Orthogonality*** – concepts should be totally independent of one another – a stronger demand than linear independence. Propriety and Orthogonality lead to modularity.

The **2nd** knowledge frame, *Quantum theory* is a generic frame, explaining "***whole systems*** in terms of their ***parts***". It is applicable to a wide variety of physical systems and their parts, e.g. crystals, molecules, atoms, and particles.

Quantum Software Models (QSM) were inspired by *Quantum Computing,* originally proposed by the physicist Feynman [11] to perform challenging computations simulating physical ***systems composed of particles***. The QSM novelty is the application to ***whole software systems*** in terms of ***modules***.

A. From Software Modules to a Whole Software System and Back

A Schematic Transition Diagram outlining a path from separate module Density Matrices, through a Direct Sum, to a whole software system Density Matrix, and back, is in Fig. 1.

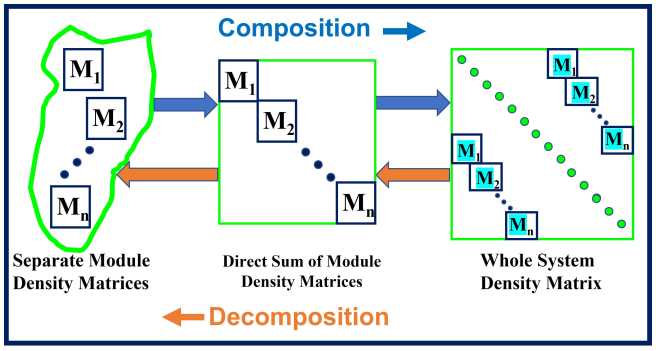


Figure 1. Schematic Transition – from separate module Density Matrices to a whole Software System Density Matrix, and back, with the intermediate modules Direct Sum. Composition is from left to right (blue arrows). Decomposition is from right to left (orange arrows). (Figures online in color).

The direct sum is formally defined for a few mathematical objects. This work only refers to matrices and subspaces *.

A matrix direct sum [20] – with symbol $\oplus$ – of n square matrices $(M_1, M_2,\ldots,M_n)$ with possibly different sizes, constructs (see Fig. 2) a block diagonal matrix as follows

$$\oplus_j M_j = \mathrm{diag}(M_1, M_2,\ldots,M_n) \quad (1)$$

DOI: 10.18293/SEKE2022-158

* Direct sums are also defined for mathematical "modules", which are different entities from this paper's *software modules*.

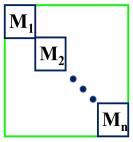


Figure 2. Matrices Direct Sum – constructs an enclosing block-diagonal Matrix.

A subspaces direct sum refers to subspaces having only the zero vector in common. As Density *Matrices* act in Hilbert *spaces* (e.g. [17] page 66), their direct sums, for software matrices and their sub-spaces, are effectively equivalent.

B. *Paper Organization*

The remaining of the paper is organized as follows. Section II reviews the basics of software system algebraic representation. Section III illustrates the Direct Sum composition of modules into a Classical Software System, then formulates the software modules' direct sum procedure. Section IV illustrates the procedure for a Quantum Software System. Section V validates the direct sum procedure from a few viewpoints. Section VI is a concise review of related works. Section VII concludes the paper with a discussion.

II. THE BASICS OF SOFTWARE SYSTEM ALGEBRAIC REPRESENTATION

A. *From Bipartite Graph through Laplacian to Density Matrix*

Software system algebraic design refers to 3 entities:

- *Structors* are generalizations of classes in Object Oriented Design (OOD).
- *Functionals* generalize OOD class methods.
- *Concepts* impart meaning to software systems' Structors and Functionals.

The relations between Structors labelled by (S1, S2,…,Si), and respective provided Functional declarations labelled by (F1, F2,…,Fk) are depicted in bipartite graphs [23] as in Fig. 3.

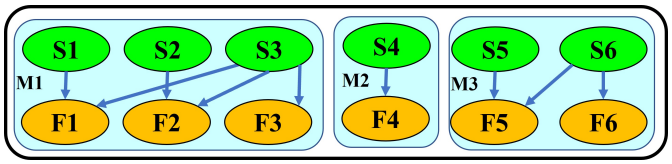


Figure 3. Bipartite Graph of a Command Design Pattern – It has 12 vertices, 6 Structors (***Si***), 6 Functionals (***Fk***), decomposable into 3 modules (***Mj***) (blue background). For instance, Structor S6 provides Functionals F5, F6 as shown by arrows. See section III-*A* for the conceptual meanings of the Command Design Pattern Structors and Functionals.

The Laplacian Matrix $\boldsymbol{L}$ is defined upon a graph as follows:

$$\boldsymbol{L = D - A} \tag{2}$$

$\boldsymbol{D}$ is the diagonal Degree matrix with $\boldsymbol{D_{ii}}$ the vertex $\boldsymbol{i}$ degree, and $\boldsymbol{A}$ is the Adjacency matrix with negative 1-valued $\boldsymbol{A_{ij}}$ if vertex $\boldsymbol{j}$ is a neighbor of vertex $\boldsymbol{i}$, and zero-valued otherwise.

Von Neumann's ([22], pages 194, 214) Density Matrix $\boldsymbol{\rho}$ – the cornerstone of our *Quantum Software Models*, representing software systems – is easily obtained from a Laplacian, (Braunstein et al. [2]), by normalizing the Laplacian Matrix by its Trace (sum of diagonal degrees) $\boldsymbol{Tr(L)}$:

$$\boldsymbol{\rho = L / Tr(L)} \tag{3}$$

B. *Software Modules Represented by Density Matrices*

A Density Matrix $\boldsymbol{\rho}$ may represent whole software systems or their modules. For example, the rightmost module M3 in Fig. 3, is shown again in Fig. 4, beside its Density Matrix. The Density Matrix – by eq. (3) – is the Laplacian Matrix (within parentheses) normalized by the factor 1/6.

A Density Matrix, as a normalized Laplacian, preserves all Laplacian properties, such as rows/columns summing zero, positive diagonal, and so on.

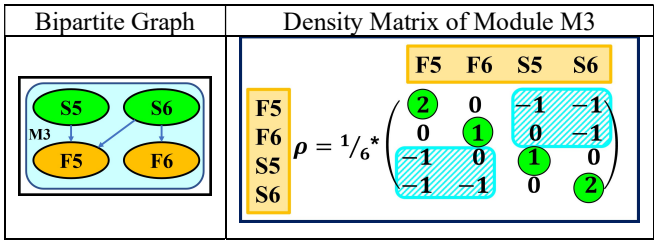


Figure 4. Bipartite Graph and Density Matrix $\boldsymbol{\rho}$ of module M3 – The left panel bipartite graph shows 4 vertices, 2 Structors labelled (**S5, S6**) and 2 Functionals labelled (**F5, F6**) as in Fig. 3. The right panel shows the Laplacian (within parentheses) {by eq. (2)}. The 4 columns and 4 rows have the same vertex order (light orange). The Degree matrix $\boldsymbol{D}$ is the diagonal (green circles). The Adjacency matrix $\boldsymbol{A}$ (hatched blue) is in the upper-right and lower-left quadrants. The Density Matrix $\boldsymbol{\rho}$ is the Laplacian normalized by the factor 1/6 {eq. (3)} since the sum of degrees of the Diagonal $\boldsymbol{D}$ equals 6.

III. FROM MODULES TO SOFTWARE SYSTEM: BACK & FORTH

This section, after an introductory classical software system, describes procedural tasks for composing a set of well-designed software modules, taken as sub-systems of a whole software system. The outcome is a formal procedure with an intermediate direct sum of modules.

A. *Introductory Classical Software System: Command Pattern*

Design Patterns are canonical software sub-systems to be used and re-used. Four authors, known as the *Gang-of-Four*, collected patterns in the "*GoF*" book [13].

The Command Pattern is an example, whose basic ideas are: a- it is an abstraction applicable to any common command – copy, paste, delete, save, etc. b- it has four *generic concepts*: an ***invoker***, e.g. a menu-item or a button; a chosen ***command***, one of the previously mentioned, e.g. copy; a ***receiver*** of the command application, e.g. a file or a document; a history mechanism, enabling ***undo***/***redo*** of commands perhaps invoked by mistake. These concepts, Structors & Functionals of this software system, assign conceptual meanings to the ***Si*** and ***Fk*** labels of Fig. 3. Command Pattern Structors & Functionals are shown together with their modules, in Fig. 5.

	Modules	Module Size		Structors		Functionals
M1	Command	3-by-3	S1	*I*Command	F1	Execute
			S2	*I*History	F2	Undo/Redo
			S3	Concrete Command	F3	Specify Command
M2	Invoker	1-by-1	S4	Action Invoker	F4	Invoke
M3	Receiver	2-by-2	S5	*I*File Receiver	F5	Receiver Action
			S6	Concrete Receiver	F6	Specify Receiver

Figure 5. Command Design Pattern: Structors & Functionals – It has 3 modules: Command, Invoker, Receiver. Compare with bipartite graph in Fig. 3. Structors' names with an *"**I**"* are *Interfaces* inherited by *Concrete* classes.

Next are transition stages from modules (Fig. 6), through a Direct Sum (Fig. 7) to System Density Matrix (Fig. 8).

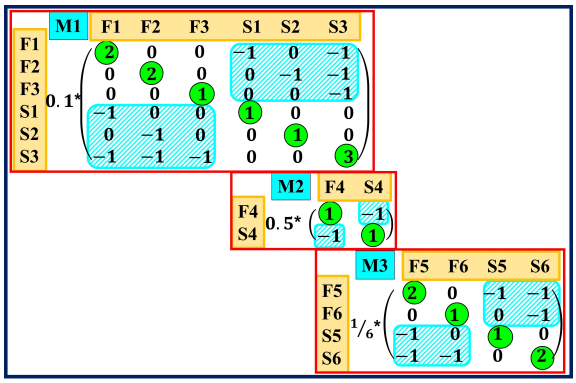


Figure 6. Command Design Pattern: Separate Modules – One sees 3 separate module Density Matrices (within red rectangles), and different normalizations: M1 has a factor 1/10, M2 has a factor ½ and M3 has a factor 1/6.

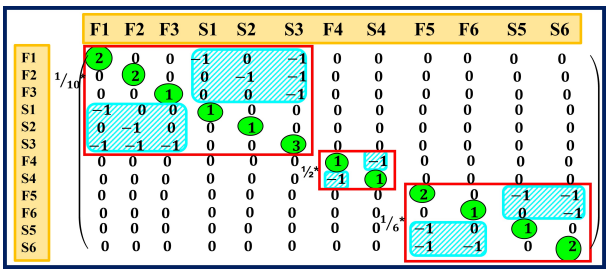


Figure 7. Command Design Pattern: Direct Sum of Module Density Matrices – The single big matrix has 3 block-diagonal modules within red rectangles.

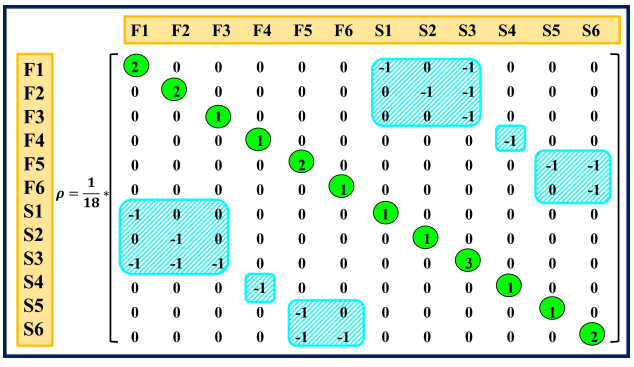


Figure 8. Command Design Pattern: Renormalized whole Software System Density Matrix – The diagonal Degree Matrix ***D*** (green circles) keeps the Direct Sum values, reordered due to repositioned Functionals and Structors. The Adjacency Matrix ***A*** (hatched blue) keeps values as each module carries the same Direct Sum columns and rows. Modules order is preserved.

Underlying the Transition from modules to the whole software system 3 tasks were performed:

1- ***Reordering Structors & Functionals*** – composition collects Structors together and Functionals together; decomposition separates them by modules;
2- ***Preserving modules order*** – the stage from Direct Sum to whole System Density Matrix, and vice-versa, keeps the same modules order;
3- ***Renormalization*** – transition through direct sum needs renormalization of density matrices in both directions.

B. Composition Procedure by Direct Sum of Modules

Here the Composition Procedure from modules to a whole software system is formulated in pseudo-code.

Composition Procedure 1– by Direct Sum of Modules
Given: Density Matrices of all Modules needed;
Obtain: Density Matrix of the whole Software System.

Preparation Phase
1. *Modules Choice* – Choose desired modules;
2. *Normalization factors* – Delete from matrices: they are not needed for composition;

Direct Sum Phase
1. *Modules' order Choice* – Decide the modules order, to be preserved in the Transition Phase;
2. *Prepare Direct Sum* – in block-diagonal format, in the decided modules' order;

Transition to Whole System Density Matrix
1. *Prepare labels list* – first Functionals, then Structors;
2. *Prepare empty system Density Matrix* – with size equal the sum of modules' Density Matrices' sizes;
3. *Loop* on list of modules, preserving their order;
   For each module Density Matrix do:
   - Add upper-right Adjacency matrix square module (including its zeros) to the system Density matrix according to the module respective row and column labels, keeping the block diagonal format of the upper-right quadrant;
   - Add diagonal degrees in the same labelled rows, such that the row values sum to zero;
   - Reflect the Adjacency matrix module, around the diagonal, to the lower-left quadrant;
   - Fill-in remaining empty system matrix elements of the whole system density matrix with zeros.
4. *Renormalize the system Density Matrix*
5. *Output system Density Matrix*

The reverse Decomposition Procedure from the whole system Density Matrix back to modules' Density Matrices is easily inferred from the above Composition Procedure.

IV. Quantum Software System and Other Systems

Having illustrated the *classical* Design Pattern, here we describe a *quantum* software system, viz. Grover Search. In addition, we concisely mention other systems.

A. *Quantum Software System: Grover Search*

Quantum Software systems design starts getting Structors & Functionals from Quantum Circuits ([17] page 22), i.e. sequential circuits with time increasing from left to right.

The Grover quantum Algorithm [14] searches unsorted databases with quadratic speedup relative to classical algorithms. Grover search begins with equal probability qubits superposition by the Hadamard operator ***H*** to the tensor power of ***n***, ending with measurement. Next are quantum circuit, its Direct Sum, and system Density Matrix (in Figs. 9, 10, 11).

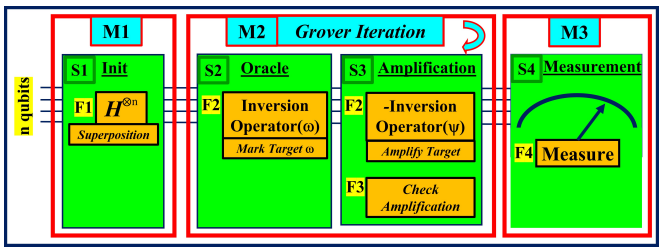


Figure 9. Grover Search Quantum Circuit – It has four (green) "boxes", the system Structors {S1,S2,S3,S4}. Each Structor contains one or more Functionals {F1,F2,F3,F4}. The Amplification Structor S3 has two Functionals F2 and F3. Modules {M1,M2,M3} (in red rectangles) contain one or more Structors. The Grover Iteration module is a loop executed alternating the Functionals inside the Oracle S2 and Amplification S3 Structors. Check Amplification F3 decides when the loop ends, passing results to Measurement.

The *Inversion Operator* F2 of the Grover Iterator (Fig. 9) is used in both Iterator Structors. F2 has the form $\boldsymbol{I - 2^*|x\rangle\langle x|}$ ([17] page 251) with $\boldsymbol{x}$ the correct searched item value $\boldsymbol{\omega}$ in the Oracle S2, or any item $\boldsymbol{\psi}$ in the Amplification S3, where F2 is multiplied by a minus sign. This is a typical inheritance case between Structors, similar to classical software inheritance.

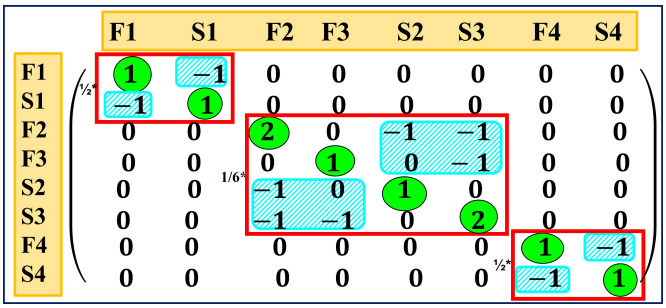


Figure 10. *Grover Search* Direct Sum of Modules' Density Matrices – modules are block-diagonal, enclosed by red rectangles.

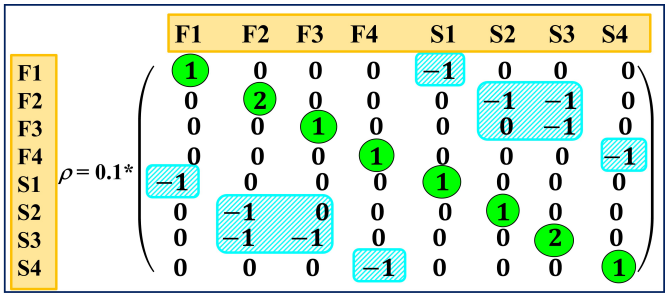


Figure 11. *Grover Search* whole software system Density Matrix.

Let us compare the Command Design Pattern – the classical software system – with the Grover Search – the quantum software system.

The diagrams serving as information source for the whole system and its modules, are different:

- UML class diagrams for classical systems;
- high-level quantum circuits, such as Fig. 9, for quantum systems.

However, once one has the list of Structors & Functionals, from then on, the procedure is identical, and totally independent of the conceptual meanings of the system.

This observation reinforces the plausibility of the idea that the same design approach should be applied to whatever kind of software system, classical, quantum or hybrid.

B. *Hybrid and Simulation Systems*

We have performed studies of other systems, which due to space limitations are not shown here. They will appear in a longer paper to be published. Other studies include: a) hybrid systems containing classical and quantum sub-systems; b) software systems whose purpose is not to compute specific numerical/logical results; the ultimate purpose of these software systems is to simulate real-world systems, to enable verification of the correctness of their control mechanisms, e.g. elevator systems.

V. VALIDATION

This paper's claim, the Software Density Matrix is a Perfect Direct Sum of Modules' Density Matrices, and its implications are formally validated here, from three complementary viewpoints:

- The meaning of ***Perfect*** Direct Sum;
- ***Density Matrix*** as a complete information source of the Software System;
- Software ***Conceptual Integrity*** from Modularity.

A. *Meaning of Perfect Direct Sum*

We first provide a definition of a well-designed software system, in terms of algebraic Conceptual Integrity, in the next text-box.

Definition 1 – Well-designed Software System
A Density Matrix software system, well-designed in algebraic Conceptual Integrity terms, obeys the following Adjacency Matrix conditions, within the Density Matrix:

- *Linear Independences* – all its Structors are mutually linearly independent and all its Functionals are mutually linearly independent;
- *Square Adjacency Matrix* – each quadrant of the Adjacency Matrix, within the Density Matrix is square, a linear algebraic consequence of the previous Linear Independences' condition [6];
- *Orthogonality* – all modules in both quadrants of the Adjacency Matrix, are block diagonal.

The meaning of the Perfect Direct Sum of the modules composing a software system is formulated in the next theorem.

Theorem 1 – *Perfect* Direct Sum of Software System Modules' Density Matrices

Assuming:

(a) each module Density Matrix of a chosen set of modules is well-designed,

and:

(b) The whole software system Density Matrix, composed of the chosen set of modules, is obtained from the Direct Sum of the modules' Density Matrices by the Composition Procedure 1;

Then:

a- The composed whole software system Density Matrix is well-designed;

b- The composed whole software system Density Matrix ***perfectly*** contains all the information of the Direct Sum, no less and no more, in the following sense:

1. The ***degrees'*** diagonal of the whole system Density Matrix has exactly the same matrix elements of the Direct Sum diagonal, only reordered;
2. The modules in the upper-right quadrant of the whole system Density Matrix have exactly the same matrix elements of the Direct Sum modules, in exactly the same modules' order;
3. The ***renormalization factor*** of the whole system Density Matrix – the ***inverse of the sum-of-degrees*** – is the inverse of the sum of denominators of the normalization factors of the Direct Sum modules.

Proof:

Item a- all vectors within each module are linearly independent by the assumption (a) of well-designed modules; modules block-diagonality follows from their matrix elements being in disjoint differently labelled columns and rows; the overall modules constitute a square, since their number of columns equals their number of rows;

Item b- 1. The degrees diagonal matrix elements are generated for all rows of each module, the same rows of the Direct Sum; they are reordered, since the columns/rows are repositioned;

Item b- 2. Exactly the same matrix elements in the same modules' order is determined by assumption (b) the Composition Procedure 1;

Item b- 3. The normalization factors' denominators are the sum-of-degrees of the respective Density Matrix. By the *Item b- 1* the degrees' of the whole system Density Matrix are exactly all the Direct Sum diagonal elements. Thus the whole Density Matrix denominator is the sum of the modules sum-of-degrees. □

B. Density Matrix: Complete Source of Software System Information

Here we regard the whole software Density Matrix as an information source about the software system, and inquire about its information completeness.

Composing the software system from modules, through the modules' Direct Sum, two information aspects need analysis:

- Completeness ***about each module*** – Theorem 1 is a full positive answer: no information is lost in the process from separate modules, through Direct Sum, to a whole system.
- Completeness ***beyond individual modules*** – the software representation of Procedure 1 is a Density Matrix for all purposes. The 1st quantum computing axiom, in von Neumann's Density Operator version ([17] page 102) is a full positive answer: "the system is completely described by its density matrix acting on the system state space".

C. Software Conceptual Integrity from Modularity

Modularity is an essential contribution to Conceptual Integrity of the whole system Density Matrix, obtained by Composition Procedure 1. This follows from assumption (a) of the *Perfect* Direct Sum Theorem 1.

Definition 1, on well-designed software systems, the basis of the referred assumption (a), explicitly incorporates the linear algebraic expressions of Frederick Brooks' underlying principles of Conceptual Integrity, viz. *Propriety* and *Orthogonality* [4].

VI. Related Works

This section offers a very concise review of related works.

We start with algebraic approaches to modularity. Within *Linear Software Models*, two spectral approaches to modularity were developed: one used the Modularity Matrix [7] and another used the Laplacian Matrix [8]. In both cases, modules were extracted from matrix eigenvectors. Within *Quantum Software Models*, Exman andrefer Shmilovich modularized software system Density Matrices [10] based upon disjoint projectors of the Matrix subspaces.

Non-algebraic alternatives have been often based upon a DSM (Design Structure Matrix) [21]. Some of them [5], were justified by economic arguments (Baldwin & Clark [1]). Another alternative, to extend UML to quantum circuits [18], is certainly interesting and accumulated large experience, but still lacks a rigorous and self-consistent theory. See also Rodriguez on OpenUP [19] and its bibliography.

Next, we refer to *Quantum Computing* (QC) (e.g. [17]) issues. This paper's Introduction states that QC is a foundation of *Quantum Software Models*. A QC modularity issue addressed by these models is human understanding and reuse of quantum circuits or parts thereof, as opposed to emphasis on efficient runtime implementation. An example [12] refers to alternative Grover iteration optimizations, with this respect.

Finally, we highlight the concepts' importance to software design. Conceptual Integrity notions and underlying principles were introduced in the well-known books by Frederick Brooks, *The Mythical Man-Month* [3] and *The Design of Design* [4]. A recent contribution to the software design field, is the book by Daniel Jackson "*The Essence of Software*", in which he explains why concepts matter for great design [15].

This, and our previous papers, claim that an algebraic approach is essential for software design, as discussed next.

VII. Discussion

A. Density Matrix Choice for Software Design

The first and foremost consideration for the Density Matrix choice for software design, is the Quantum theory in which it is embedded. It offers invaluable benefits: it is rigorous, self-consistent, and triggers otherwise unthinkable questions. Furthermore, the Density Matrix affords utmost generality relative to software systems' pure or mixed states.

Practical benefits of the algebraic design are:

a) ***Agile-Design-Rules*** –Brook's Conceptual Integrity [3][4], implemented by a Laplacian or a Density Matrix, is a theory behind Agile-Design-Rules, conforming to accepted best practices' wisdom (see [9] and its bibliography);

b) ***Input modules a priori correctness*** – the correctness of Direct Sum input modules, is assured by algebraic spectral modularization and its specialized decoupling techniques [8].

c) ***Format uniformity*** & ***Software generality*** – the core asset of the algebraic software design, is an extremely simple and uniform Density Matrix format, stimulating general applicability to software systems of any kind, size, or module numbers.

B. Direct Sum, Direct Product and Tensor Product

We use Direct Sum and not Direct Product because a direct sum element is nonzero only for a *finite* number of entries, and our Density Matrices act on *finite* vector spaces. Direct product elements may have an infinite number of nonzero entries [20]; as Lang notes for abelian groups ([16], page 36), the direct sum is a direct product subset.

The Tensor Product is applicable to Density Matrices. But its relation to Direct Sum, in the software design context, is out of the scope of this paper, and will be discussed elsewhere.

C. Future work

Two topics are the subject of future work:

1st- *Outliers' Presence* – the Direct Sum has been applied to modules' Density Matrix in the Composition Procedure 1, assuming absence of outliers coupling modules. Outliers will be pre-processed by module decoupling techniques [8].

2nd- *Overlapping concepts* – a Natural Language post-processing function will be developed, to check the existence, and to streamline semantically overlapping concepts, occurring in different modules composed by Procedure 1.

D. Main Contribution of this Paper

This paper's main theoretical contribution is the Module Matrices' Direct Sum as a perfect composition of the whole software system Density Matrix. Practical benefits are design flexibility, enabling whole system gradual build-up from verified correct modules, assured by the theoretical framework.

References

[1] Carliss Y. Baldwin and Kim B. Clark, *Design Rules*, Vol. I. The Power of Modularity, MIT Press, MA, USA, 2000.

[2] Samuel L. Braunstein, Sibasish Ghosh and Simone Severini, "The Laplacian of a graph as a density matrix: a basic combinatorial approach to separability of mixed states", https://arxiv.org/abs/quant-ph/0406165 Oct 2006.

[3] Frederick P. Brooks Jr., *The Mythical Man-Month – Essays on Software Engineering* – Anniversary Edition, Addison-Wesley, Boston, MA, USA, 1995.

[4] Frederick P. Brooks Jr., *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, Boston, MA, USA, 2010.

[5] Yuanfang Cai and Kevin J. Sullivan, "Modularity Analysis of Logical Design Models", in *Proc. 21st IEEE/ACM Int. Conf. Automated Software Eng. ASE'06*, pp. 91-102, Tokyo, Japan, 2006.

[6] Iaakov Exman, "Linear Software Models: Standard Modularity Highlights Residual Coupling", Int. Journal on Software Engineering and Knowledge Engineering, vol. 24, pp. 183-210, March 2014. DOI: 10.1142/S0218194014500089

[7] Iaakov Exman, "Linear Software Models: Decoupled Modules from Modularity Matrix Eigenvectors", Int. Journal on Software Engineering and Knowledge Engineering, vol. 25, pp. 1395-1426, October 2015. DOI: 10.1142/S0218194015500308

[8] Iaakov Exman and Rawi Sakhnini, "Linear Software Models: Bipartite Isomorphism between Laplacian Eigenvectors and Modularity Matrix Eigenvectors", Int. Journal of Software Engineering and Knowledge Engineering, Vol. 28, No 7, pp. 897-935, 2018. DOI: http://dx.doi.org/10.1142/S0218194018400107

[9] Iaakov Exman, "Conceptual Software: The Theory Behind Agile-Design-Rules", in Proc. SEKE'2018 30th Int. Conf. on Software Engineering and Knowledge Engineering, Redwood City, CA, USA, pp. 110-115, July 2018. DOI: 10.18293/SEKE2018-182.

[10] Iaakov Exman and Alon Tsalik Shmilovich, "Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design", (2021) https://arxiv.org/abs/2103.13755

[11] Richard P. Feynman, "Simulating Physics with Computers", Int. J. Theor. Phys., 21:467, 1982.

[12] Caroline Figgatt, Dmitri Maslov, Kevin A. Landsman, Norbert M. Linke, Shantanu Debnath and Christofer Monroe, "Complete 3-Qubit Grover Search on a programmable quantum computer", Nature Communications, 8: 1918, 2018. DOI: https://doi.org/10.1038/s41467-017-01904-7

[13] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston, MA, 1995.

[14] Lov K. Grover, "Quantum Computers can search arbitrarily large databases by a single query", Phys. Rev. Lett. 79(23): 4709-4712, 1997.

[15] Daniel Jackson, *The Essence of Software*, Princeton University Press, Princeton, NJ, USA, 2021.

[16] Serge Lang, *Algebra*, Revised 3rd edition, Springer-Verlag, Berlin, 2002.

[17] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[18] Ricardo Perez-Castillo, Luis Jimenez-Navajas and Mario Piattini, "Modelling Quantum Circuits with UML", in Proc. QSE'2021 Quantum Software Engineering Workshop. https://arxiv.org/abs/2103.16169.

[19] Andres Rodriguez, "Extending OpenUP to Conform with the ISO Usability Maturity Model", in Sauer et al. (eds.) *Human-Centered Software Engineering*, HCSE 2014, LNCS, vol. 8742, Springer, Berlin, pp. 90-107, (2014). https://doi.org/10.1007/978-3-662-44811-3_6

[20] Todd Rowland and Eric W. Weisstein, "Direct Sum", https://mathworld.wolfram.com/DirectSum.html, 2022.

[21] Neeraj Sangal, Ev Jordan, Vineet Sinha and Daniel Jackson, "Dependency Models to Manage Complex Software Architecture", Proc. *OOPSLA'05*, pp. 167-176, October 2005. https://doi.org/10.1145/1094811.1094824

[22] John von Neumann, *Mathematical Foundations of Quantum Mechanics*, New Edition, Princeton University Press, Princeton, NJ, USA, 2018.

[23] Eric W. Weisstein, Bipartite graph (2022), http://mathworld.wolfram.com/Bipartite-Graph.html

[24] Eric W. Weisstein, Laplacian (2022), http://mathworld.wolfram.com/LaplacianMatrix.html

SCMA: A Lightweight Tool to Analyze Swift Projects

Fazle Rabbi
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
bsse0725@iit.du.ac.bd

Syeda Sumbul Hossain
Department of Software Engineering
Daffodil International University
Dhaka, Bangladesh
syeda.swe@diu.edu.bd

Mir Mohammad Samsul Arefin
Information Engineering
Chalmers University of Technology
Gothenburg, Sweden
s.arefin@outlook.com

***Abstract*—In global software engineering, practitioners use code metrics analyzers to measure code quality to detect code smells or any technical debt early at the development phase. Different tools exist to evaluate these metrics to ensure the maintainability and reliability of any codebase. This paper presents a tool SCMA (Swift Code Metrics Analyzer) which analyzes swift code considering ten code metrics for analyzing software architecture to ensure code quality. We have used the native swift parser to implement this tool. This tool suggests refactoring the codebase by giving a final score averaging the score of all ten metrics. We have validated the accuracy of each metric measured by this tool by analyzing the codebase manually. This tool can help the developers to inspect the swift modules of iOS projects and give an insight into the improvement area of each project.**

***Index Terms*—Code metrics, Code Metrics Analyzer, Swift Language, Code Quality, Code Smell**

I. Introduction

Nowadays, software companies are evaluating their projects in terms of different software metrics. These metrics are categorized into three different types- Process metrics, Project metrics, and Product metrics [3]. These metrics are evolving to measure the software development processes to enhance the development models. Numerous studies further have been done on these metrics. A systematic mapping study [4] has been done on software process metrics where process metrics had categorized into three different types. Software Product metrics are also studied over time which is stated in another mapping study [1]. Among those metrics, product metrics define product quality. Source code metrics are one of the product metrics that helps to measure the code quality of any codebase [8]. There are around 300 code metrics found [8] in different literature.

Clean architecture becomes a buzzword among the software engineering practitioners motivated by R. Martin (uncle Bob) [7]. To ensure the code quality, architectural analysis of the codebase is crucial to avoid code smells or any other technical debts. There are different types of code smells are stated in different literature. These code smells tends to code refactoring [2].

This paper presents a tool SCMA (Swift Code Metrics Analyzer) which analyzes swift code considering ten code metrics for analyzing software architecture to ensure code quality.

The rest of the paper is organized in accordance: Background Study at Section II followed by Tool Description and Discussion at Section III and Section IV respectively. We have reviewed the validity of our study in Section V. Finally Section VI furnishes our contribution.

II. Background Study

A. Technical Debt (TD)

Technical debts are short-term benefits that occurred by the software engineers unintentionally throughout the software development processes. There are different types of technical debts stated in studies. A systematic mapping study [6] had been done on 94 studies where technical debt is classified into 10 different types as Requirements TD, Architectural TD, Design TD, Code TD, Test TD, Build TD, Documentation TD, Infrastructure TD, Versioning TD, Defect TD. This paper also stated that Code TD had studied most.

B. Code Metrics

Code metrics are being used to get the insight of source code written by the developers in terms of some measurement units. As these metrics reveal the code health of any codebase at the development phase, practitioners can easily get the idea of improvement areas of their codebase. Considering the nature of projects and processes in global software development, practitioners are using different code metrics. Kitchenham, B. [5] had conducted a preliminary mapping study on software metrics. A systematic mapping study [8] has been done on software code metrics where a total of 226 studies have been gone through to map almost 300 code metrics of 13 different programming languages (Java, AspectJ, C++, C, C#, Jak, Ada, Cobol, Javascript, Pharo, PHP, Python, and Ruby). This study also summarizes 41 different software metrics tools into two categories: commercial tools which are free or paid tools and others that are developed by the authors. Another systematic mapping study [9] had been performed on dynamic software code metrics which illustrated that coupling, cohesion, complexity, method invocation, memory allocation and usages were mostly focused research topics.

DOI reference number: 10.18293/SEKE2022-006

C. Tool used within our organization

In our organization we are using our own tool, to measure the different code metrics associated with their organization. This tool summarizes a score using the code metrics of LOC (Line of Code), Global Variables, Predefined processors, Cyclomatic Complexity, Duplicate Code, and Modular Circular Dependency. This tool supports projects with languages like Java, C, C++, C# excluding iOS supported languages, especially swift. To be aligned with the organization's coding culture, we are motivated to develop a code metrics analyzer that supports swift language.

D. Code Metrics Tools for Swift

Swift language has evolved later in 2014 and becomes more popular for iOS development. Clean Swift [1], a set of rules, is introduced with XCode for maintaining better architecture for swift projects. As per the best of our language, there are a few software code metrics tools that exist for swift language which measures some basic code metrics. SonarSource [2] is a static analyzer that checks 119 predefined rules for swift language which covers unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in SWIFT code. It also supports other 26 languages and has specific rule sets based on the languages.

Code Climate [3] is a repository-based metric analyzer for swift language that checks duplication, cognitive and cyclomatic complexity and some others basic checks.

Swift Code Metrics is another tool for swift projects [4] by which some basic code metrics (the overall number of concrete classes and interfaces, the instability and abstractness of the framework, the distance from the main sequence, LOC (Lines Of Code), NOC (Numbers Of Comments), POC (Percentage Of Comments), NOM (Number of Methods), Number of concretes (Number of classes and structs), NOT (Number Of Tests), NOI (Number Of Imports), and Frameworks dependency graph (number of internal and external dependencies)) can be analyzed.

Taylor [5] is another tool for analyzing swift code which considers the code metrics Excessive Class Length, Excessive Method Length, Too Many Methods, Cyclomatic Complexity, Nested Block Depth, N-Path Complexity, and Excessive Parameter List. SwiftLint [6] checks over 200 rules, including 12 code metrics for swift languages.

III. Tool Description

In this section, we presented the overall activities of our tool from parsing source codes to generating html reports. The score calculation method from metrics is also shared.

[1]https://clean-swift.com/
[2]https://rules.sonarsource.com/
[3]https://codeclimate.com
[4]https://github.com/matsoftware/swift-code-metrics
[5]https://github.com/yopeso/Taylor
[6]https://github.com/realm/SwiftLint

A. SOURCE CODE PARSING

At the very beginning, the source code files with .swift extensions are selected from a project. The contents are read one by one and converted into Abstract Syntax Trees (AST). To parse these files and AST preparation, SwiftSyntax [7] is used as a parser. Each of the AST contains a class, methods under the class, global variables and variables under methods, and other elements from a source code file as branches in hierarchy order. Figure 1 illustrates the parsing procedure of source codes.

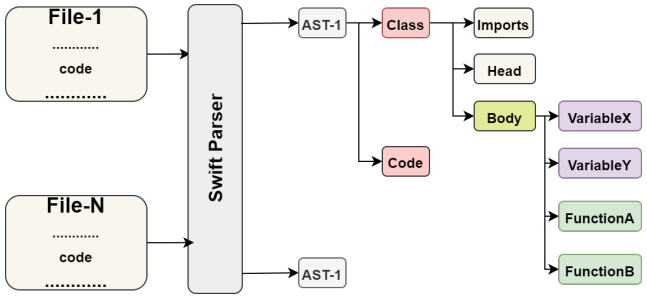


Fig. 1. Source Code Parsing

B. METRICS DETAILS

After ASTs are generated from source code files, information such as class names, lines of codes, method names, variables names are collected from the ASTs. From that information, a call graph matrix is prepared for a project where every node (methods, variables) is linked to other nodes with whom they have a connection. Each of the properties as well as each of the relations in a project is considered to calculate metrics. After this step, the following metrics are captures. For every metric, a violation count is considered. To keep the project well managed and keep the score good, violations must be avoided. The metrics to be considered with their brief details in SCMA tool are as follows:

1) **Line of Code by Classes (LOCC):** Total line of code without comments in a class are considered the first information to be used as a metric. More lines in a class hampers the maintaining activities and we considered it as a negative impact to our score calculation.
2) **Weighed Method Count by Classes (WMCC):** Summation of Cyclometic complexities of all methods in a class. Classes with high Weighted Method Count have less readability.
3) **Number of Methods by Classes (NOMC:** Count of methods in a class. More methods in a class increases the complexity in a class. The number of methods should be kept low.
4) **Number of Global Variables by Classes (NOGC):** Count of global variables in a class. Number of Global Variables must be kept as low as possible to make a good score by the tool.

[7]https://github.com/apple/swift-syntax

5) **Number of Couplings by Classes (NOCC):** A Coupling is considered when two classes have at least one mutual connection in any direction. This metrics represents the number of total couplings in a project. To maintain a good readability, couplings should not be presented very high between classes. Low number of couplings will help to make a good score.
6) **Number of Accessed Methods by Variables (NOAV):** Total number of accessible global variables inside a method. More usage of global variables by methods increases the cohesion and contributes to the score.
7) **Line of Code by Functions (LOCF):** Total line of code in a method body. As like LOC for classes, LOC for function also should be kept as low as possible.
8) **Cyclomatic Complexity by Functions (CCF):** The quantitative measure of the number of linearly independent paths through a program's source code. To keep good readabilities, cyclomatic complexity must be small in number.
9) **Number of parameters by Functions (NOPF):** Number of total parameters in a method signature. It should be kept as low as possible.
10) **Duplicate Code (DC):** Same lines of code between two or more methods/class/blocks introduces code duplicacy. Duplicate codes are generated by copying and pasting codes and they introduce bugs while editing the codes later. In our tool we use lizard[8] to count the duplicate blocks of codes.

The overall procedure of this step is shown in Figure 2

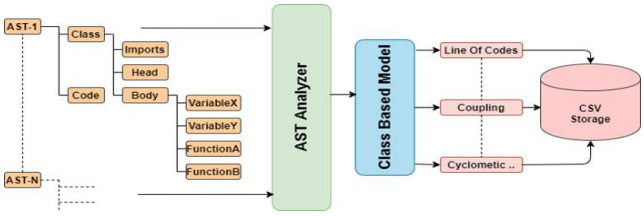


Fig. 2. Metrics Calculation

C. SCORE CALCULATION

In this section, the score calculation from metrics is described briefly. Table III-C illustrates the details of violations and formula we used for score calculation. From each of the metrics we get a score ranging from 0 to 5. Using all of these scores, a final score is calculated by averaging the values of these scores. Alike the individual scores from metrics, the final score is also remains from 0 to 5. Score close to 5 represents the project as good conditioned close to 0 means the project contains smells which needs to be refactored.

D. REPORT GENERATION

At the final step after metrics calculation, a report needs to be generated. To generate a report, an HTML page is generated through a python server. In that report, the summary of every metrics is illustrated. Users can also get the details from a CSV file for the corresponding metric. From each of the metrics (See Figure 3), a score (0 to 5) is calculated from the violation count for few metrics.

After that, an overall score is calculated using the average value from all of the metrics scores. Figure 4 illustrates an output of the overall score.

[8]https://github.com/terryyin/lizard

TABLE I
SCORE CALCULATION FROM METRICS

Metrics	Violation	Score
Line of Code by Classes	over 500 lines	Score = $\frac{1000}{maxLOC+1500}$
Weighed Method Count by Classes	over 200 count	R = $\frac{maxWMC*violations}{totalWMC}$; Score = $\frac{2}{R+0.4}$
Number of Methods by Classes	N/A	Score = $\frac{500}{maxmethodsinaclass+75}$
Number of Global Variables by Classes	N/A	Score = $\frac{500}{maxGlobalsinaclass+75}$
Number of Couplings by Classes	N/A	Score = $\frac{1000}{maxcoupling+150}$
Number of Accessed Methods by Variables	N/A	Score = $\frac{2}{methods/globals+0.4}$
Line of Code by Functions	N/A	Score = $\frac{2000}{maxLOC+300}$
Cyclomatic Complexity by Functions	over 20 count	R = $\frac{maxCC*violations}{totalCC}$; Score = $\frac{2}{R+0.4}$
Number of parameters by Functions	over 10 params	R = $\frac{maxParm*violations}{totalParms}$; Score = $\frac{2}{R+0.4}$
Duplicate Code	over 10 lines	R = $\frac{duplicatedlines*totallines}{totalParms}$; Score = $\frac{2}{R+0.4}$

IV. DISCUSSION

The proposed tool (SCMA) is developed considering the standard of global SW engineering tools, internal organiza-

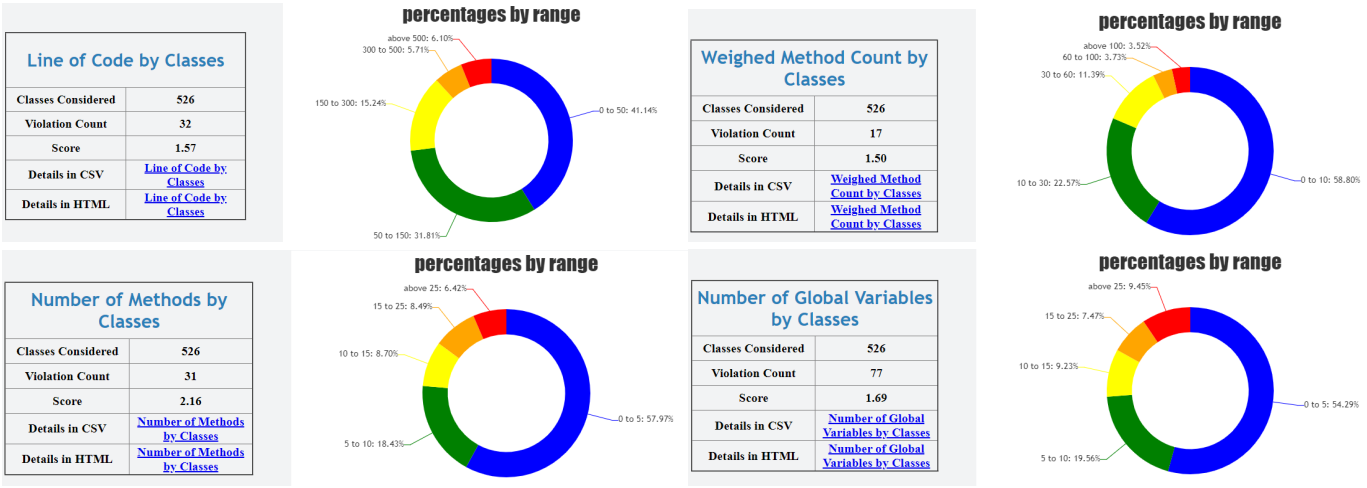


Fig. 3. Individual Metrics Score

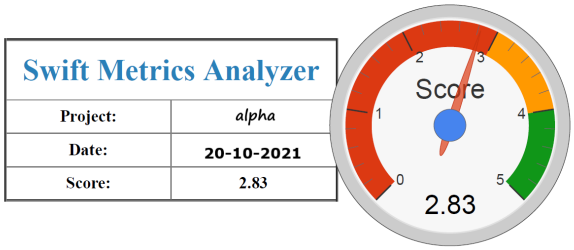


Fig. 4. Final Score

tion's guildlines and expert opinions. There is some resemblance and dis-resemblance exit among the metrics calculated by different tools. Modular Circular Dependency (MCD) and Predefined Preprocessor Metrics are not considered in this tool as they are not compatible with swift. Table II illustrates SCMA tool specification.

TABLE II
SCMA TOOL

Key	SCMA
Language	Swift
Scoring	0-5
Build Dependency	No
Report	CSV, HTML

V. Threats to Validity

a) Internal Validity: We have built this tool based on the code metrics that are being considered within our company.

b) External Validity: For analyzing the result of our tool, we have not considered any other open-source projects as we could not trace the changes in metric scores.

c) Construct Validity: We considered the scoring formula and threshold values used as standards of global software engineering and our company. After having the scores from every sections, we calculate the overall score by averaging the individual scores of all of the metrics.

d) Reliability: We have run our tool on 11 iOS based software projects. All the applications are from real-life projects available on the app store. After the first run, we have run this tool several times after refactoring the code as per the suggestions provided by our SCMA tool.

VI. Conclusion

In this paper, we proposed a tool named SCMA to automatically score a swift project using ten software code metrics. We run this tool on swift-based software projects of our company and manually validated the result for some cases. To our knowledge, this tool provides valid output. The violation counts used here are chosen from the global code smell standards, our internal organization's guidelines and expert opinions. Currently, we are working with other metrics and we will adapt those in our future work.

References

[1] Colakoglu, F.N., Yazici, A., Mishra, A.: Software product quality metrics: A systematic mapping study. IEEE Access (2021)
[2] Fowler, M.: Refactoring: improving the design of existing code. Addison-Wesley Professional (2018)
[3] Honglei, T., Wei, S., Yanan, Z.: The research on software metrics and software complexity metrics. In: 2009 International Forum on Computer Science-Technology and Applications. vol. 1, pp. 131–136. IEEE (2009)
[4] Hossain, S.S., Ahmed, P., Arafat, Y.: Software process metrics in agile software development: A systematic mapping study. In: International Conference on Computational Science and Its Applications. pp. 15–26. Springer (2021)
[5] Kitchenham, B.: What's up with software metrics?–a preliminary mapping study. Journal of systems and software **83**(1), 37–51 (2010)
[6] Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. Journal of Systems and Software **101**, 193–220 (2015)
[7] Martin, R.C.: Clean architecture: a craftsman's guide to software structure and design. Prentice Hall (2018)
[8] Nuñez-Varela, A.S., Pérez-Gonzalez, H.G., Martínez-Perez, F.E., Soubervielle-Montalvo, C.: Source code metrics: A systematic mapping study. Journal of Systems and Software **128**, 164–197 (2017)
[9] Tahir, A., MacDonell, S.G.: A systematic mapping study on dynamic metrics and software quality. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM). pp. 326–335. IEEE (2012)

Anomaly Detection in Spot Welding Machines in the Automotive Industry for Maintenance Prioritization

Laislla C. P. Brandão [1] lcpb@ecomp.poli.br
Aldonso Martins-Jr [1] amoj2@ecomp.poli.br
Gabriel A. Kopte [1] gak@ecomp.poli.br
Edson Filho [1] jeaf@ecomp.poli.br
Alexandre M. A. Maciel [1] amam@ecomp.poli.br

[1] *Department of Computer Engineering*
School of Engineering, Universidade de Pernambuco
Recife, Brazil

Abstract

Based on the need of prioritization of maintenance activities in a BOSCH Spot Welding process in the automotive industry, this work aims to develop anomalous equipment selection methodologies for assisting it. The first one is proposed based on data exploration by checking every possible set of alarms of the machines. A second one is created using multiple data clustering models in order to identify machines that behave differently from the others for certain time periods. Bayesian networks were also applied to assist the identification of cause-and-effect relationships between the warning and error logs. The clustering method proved effective in identifying anomalies, which were later inspected on the shop floor.

Keywords: data mining, maintenance, spot welding, automotive industry, anomalies.

1 Introduction

The Automotive Industry are in constant process of transformation and, in order to thrive in the era of Industry 4.0, automakers need to quickly adapt in order to continuously achieve their goals and overcome challenges, increasing the lifespan of their assets and productivity through the use of technologies such as Big Data Analytics, enabling intelligent manufacturing [1].

Having in mind that maintenance activities must make the best possible use of scheduled downtime and resources to minimize its losses, it is always necessary to select which activities should be performed. In many cases, traditional maintenance policies can be satisfactory, but when maintenance and failure costs are high [2], managing maintenance using data analysis becomes a better choice.

By analyzing the data generated by the welding machines and using data mining techniques, this study plans to develop methodologies that assist in the detection of anomalous behavior patterns, to support the reduction of unscheduled stops, maintenance time and repair costs and provide a rise in the efficiency and quality of the welding process.

DOI reference number: 10.18293/SEKE2022-118

2 Background

2.1 Spot Welding

The industrial process of manufacturing an automobile has four major workshops. The one of interest for this work is the Body-in-White Shop, responsible for the construction of the car bodies by joining its metal parts.

A common process to all body-in-white shops globally is the joining of metal sheets using a technique known as Spot Welding. The equipment used, called welding gun, uses two metallic copper electrodes to apply a force of union between metallic plates creating welding spots by passing high level electrical current through them and the metal worksheets in between [3]. The heat generated by the passage of high electric current through the small section of the electrodes melts the metals of the two plates, providing the union of the parts when the material solidifies again.

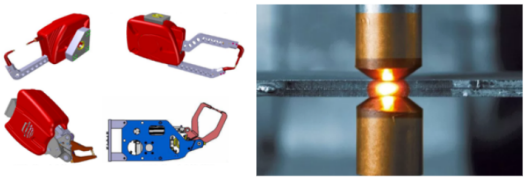


Figure 1: Example of welding guns (left) The union of two pieces of metal through spot weld (right)

A body-in-white shop contains thousands of welding guns and every machine stores information about the weld application, configuration parameters and measurements of each spot weld in a standard SQL Server database. It is available on the local manufacturing machine on the same industrial network as other welding equipment and robots.

2.2 Anomaly Detection

Cluster analysis is one of the most important research fields in data mining. Clustering belongs to the category of unsupervised learning as it does not depend on training samples, this is why they are the straightforward technique for

anomaly detection. For these algorithms, given a number of clusters k as an initial parameter, the set of data objects is divided into k categories, or groups.

An example of this type of algorithm is K-Means. One can use several sets of different starting centers for various iterative calculations and choose the best one as the final result, but one cannot guarantee that this result is the optimal solution, while several iterations consume a lot of time, a lot of uncertainty, so it is very important to select the ones suitable starting group centers [4].

Another algorithm, DBSCAN (Density-Based Spatial Clustering), is a pioneering density-based algorithm. It can discover clusters of any arbitrary shape and size in databases that contain even noise and outliers, although it has some problems, such as being subject to dilemmas when deciding meaningful clusters from datasets with varying densities [5].

When the algorithm is able to minimize an error function, it is often called C-Means where c is the number of clusters, and if the classes used are using the Fuzzy technique, then it is known as Fuzzy C-Means (FCM) [6]. Its benefit is the formation of new clusters from data points that have membership values close to existing classes. Fuzzy C-Means has the advantage of being very good for problems of many dimensions.

The most popular model of association patterns between groups of items uses item set frequencies to quantify association level, but there are also the Bayesian, that use Bayes' theorem to represent knowledge. In simple cases, the structure of Bayesian networks can be defined by an expert and used to make inferences about a given problem. In other more complex applications the structure and parameters of the network can be learned [7] [8].

2.3 Related Work

Many works found propose a prioritization approach from a maintenance perspective, with data-oriented approaches and aiming at the preventive diagnosis of problems. Several methods have been developed to identify performance bottlenecks related to maintenance activities in production systems [9]. The underlying logic behind all of them lies in analyzing the machines' event log data [10].

Data mining techniques are used to detect bottlenecks in production systems, although several methods proposed focus on analytical logs referring to the process, these data do not provide diagnostic information explaining what the root causes of incidents are [11].

In one of the approaches, Bayesian networks and attribute relevance analysis are used to process a dataset of failure records of industrial machinery components, with the purpose of using the conditional probabilities generated by the networks, as well as the relevance of the rankings of criteria for creating a decision-making model [12].

At the same time, some works specifically related to spot welding technology and case studies applied to BOSCH were also identified. Due to the great variety and diversity of data collected in the welding process and which are relevant for monitoring product quality, the work of Svetashova et al. [13] reports that they find significant challenges for the modeling of machine learning algorithms and these challenges are presented in conjunction with the predictive quality monitoring model.

3 Materials and Methods

3.1 Database Description

The CRISP-DM methodology was used for providing a framework for carrying out data mining projects, regardless of the industry sector and the technology used.

As said, this database automatically stores spot welding information in specific datasets for each purpose. The *ExtError_RDS_V* dataset contains records of important events that occurs in one of the welding controllers in the process, which may have warning codes or errors. This dataset has 13 columns and the existing fields are shown in Table 1.

For this work, the focus was given to one single production line, the bottleneck line, thus the most important one, and all its welding machines. The data considered was the gathering of a month of production.

Table 1: Dataset *ExtError_RDS_V* Data Description

COLUMN	DESCRIPTION
date	Event registration date
line	Record of the production line where the robot is located
protRecord_ID	Single consecutive number (automatic value)
dateTime	Timestamp record of the moment of occurrence of the event
timerName	Unique identification of the registered event source welding machine
errorCode1	Warning or Error code event source
errorCode1_txt	Warning or Error event source - Text in selected language
errorCode2	Secondary error code
errorCode2_txt	Secondary error code - Text in selected language
isError	Flag to identify if the event is a Warning or Error
isError_txt	Literal text Warning or Error
servodynDVState	Servo status code
servodynDVState_txt	Servo Status - Text in selected language
tablename	Address of the folder on the computer where the ExtError tables are located

3.2 Data Preprocessing

The data was loaded into a NoSQL database in the Google Cloud Computing (GCP) cloud called BigQuery and the approach chosen was firstly the adaptation of the fact tables of the database in an OLAP (Online Analytical Processing) architecture, since it allows greater flexibility and performance in data analysis. Data was transformed from categorical columns into non-categorical columns from the beginning, the primary keys with ID for the dates and machines were stored separately in the Dimension tables and the new columns *id_date* and *id_machine* were used

instead of *date* and *timerName*.

White spaces at the end of the text were removed, especially in error description columns, and a mapping of the missing data in the database was carried out, these values being later filled with a symbolic value of (-1) not to impact the operation of subsequent algorithms.

In order to make possible a more in-depth analysis regarding each of the alarms present in the database, the columns *errorCode1_txt* and *errorCode2_txt* were concatenated creating a new column, *errorCode_txt*. This brings a second level of detail for each error.

After that, a new table was created, pivoting and grouping the data of *ExtError_RDS_V* by dates and machines and presenting the number of occurrences of each of the alarms of *errorCode_txt* arranged in separate columns, one for each alarm. The new table, called *ExtError_group*, has 40 columns, two of which are the date and machine ID and the others represent each of the possible alarms in the welding guns, encompassing both warnings and errors.

3.3 Exploratory Data Analysis

The analysis were initiated by looking at the distinct values for each column of the original *ExtError_RDS_V* database and creating histograms based on their categories. At first, a focus was given to the *errorCode1_txt* column in order to identify the unique descriptions of the existing alarms (errors and warnings), and after that, an analysis of occurrences of each possible alarm was performed in the new concatenated column *errorCode_txt* to verify the types of alarms most common in the whole dataset.

From the new dataset, *ExtError_group*, it is possible to analyze separately the influence that each one of the alarms has on the behavior of the machines. For instance, the error *"welding error"* with the sub-description *"cancellation by pliers movement"* considering all machines from the perspective of each date is shown in Fig. 2.

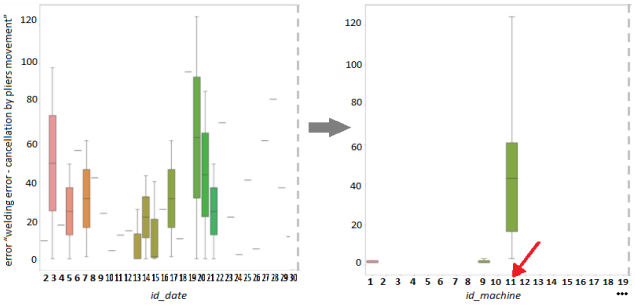


Figure 2: Variation of the error *"welding error - cancellation by pliers movement"* for each date id (left) and for each machine id (right).

When analyzing this same error from the perspective of each machines, a very important information is acquired. By looking at the boxplots, the machine id=11 stands out when compared to the others in terms of data variance, which shows that there may be an opportunity associated with this machine.

3.4 Modeling

Two approaches were used to prepare the data for the models, one using the original data from the *ExtError_group* dataset and the other using the normalized data through the StandardScaler method.

K-Means, DBSCAN and Fuzzy C-Means clustering algorithms were the techniques used for the anomaly detection. The K-Means and Fuzzy C-Means methods require the number of clusters as an initial parameter, while DBSCAN requires an agglutination radius and the minimum number of records to form a cluster. To determine these parameters, a search using the two approaches for each method and dataset was performed: Maximum Silhouette score and Elbow Curve (Inertia).

Therefore, a total of 12 models were analyzed in search of machines of interest: Three Methods (K-Means, DBSCAN and Fuzzy C-Means) × Two Datasets (Non-standardized and Standardized) × Two Approaches (Maximum Silhouette and Elbow Curve, using Silhouette for DBSCAN and Inertia for the others two methods.

Bayesian networks were used to find causality in the error and warning logs, trying to relate apparently non-relevant errors and warnings with errors related to critical failures for the machines of interest. The structure was trained using the Hill Climbing Search algorithm.

4 Evaluation and Results

4.1 Results

For the first of the 12 models, the result of the Inertia metric in the search for parameters (number of clusters) for the K-Means method and Non-standardized data is the point where a discontinuity occurs (the elbow). Here it was possible to see that the ideal number of groups that best represents the data from *ExtError_group* is four for the Elbow Curve method, showed in Fig. 3(a).

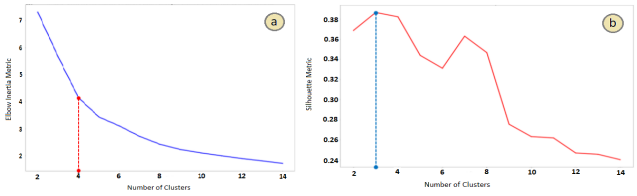


Figure 3: Result of Elbow Method Inertia vs. number of clusters (a) and Result of Silhouette Score vs. number of clusters (b).

This parameter is used to train the K-Means algorithm and then apply the model to the data to obtain an association

of each of the records in the database to one of the four clusters. This model had an overall Silhouette Score of 0.38 and an average Euclidean distance of 48.9.

Fig. 4(a) shows the number of day-machine pairs grouped in each cluster for the K-Means model with four clusters and the approach using Elbow Curve Inertia with Non-standardized data. It is observed that 28 events were isolated in group number 2, clearly different from the others grouped in large clusters. Minority groups tend to show rare and/or unusual events that may indicate good opportunities for preventive maintenance plans by characterizing machines that behaved anomalously in a small set of days.

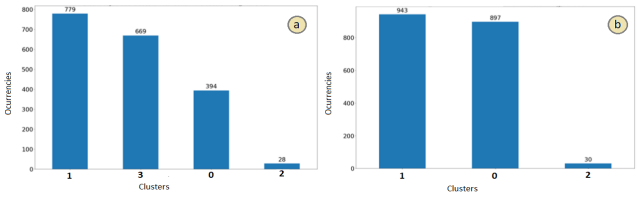


Figure 4: Distribution of clusters for K-Means model with four clusters (a) and three clusters (b) and Non-standardized data.

By looking at the records of each machine distributed among the four clusters, it was verified that the 28 events of interest occurred on the same machine, id=5. This machine was then declared a machine of interest, as it may be associated with anomalous functioning, and this information should be compared with the shop floor and the information held by the stakeholder, the welding specialist.

The Max Silhouette approach was used to select the number of clusters for training the second K-means algorithm, also applied to Non-standardized data, showed in Fig. 3(b). Three clusters were selected for this model and it was trained, obtaining the association between records and clusters. This model had an overall Silhouette Score of 0.38 and an average Euclidean distance of 41.4.

Fig. 4(b) shows the amounts of day-machine pairs grouped in each cluster of the second model, with K-Means with three clusters and the approach using Max Silhouette, and Non-standard data. It is seen that 30 events were isolated in group number 2, another notably minority group. As discussed earlier, this can characterize a set of machines with anomalous behavior.

Once again, observing the records of each machine distributed among the clusters, it was seen that the 30 events of interest in cluster number 2 are from the same machine id=5, reinforcing that this machine is a machine of interest for maintenance prioritization.

The Bayesian network was created to the machine with id=5 (the anomaly data from cluster number 2, the minority class). The resulting network was analyzed and filtered with the help of the stakeholder. Three relationships of interest were identified. Relationships 1 and 2 indicate a lack of current error related to a maximum lag warning. It refers to the phase shift of current with respect to voltage in the welding process. Relation 3, on the other hand, indicates a current oscillation problem. The main possible causes for the errors presented in 1, 2 and 3 are the same: abrasion of the welding electrode, measuring circuit or auxiliary cables; interference from other processes on the same network; and weld transformer problems (insufficient capacity).

Table 2 summarizes the result of the same methodology applied to these and the other 10 models created, and Table 3 summarizes the total occurrences of machines of interest identified by each of the models that were applied.

Table 2: Summary of algorithms, parameter selection methods, applied parameters, metrics (Silhouette Score - Distance) and the identified machines of interest.

ALGORITHM AND GROUPS	STANDARD DATA	METHOD SELECTION OF NUMBER OF GROUPS	S. SCORE AND DIST.	ID MACHINES OF INTEREST
K-Means-3	No	Max Silhouette score	0.38 - 48.9	5
K-Means-2	Yes	Max Silhouette score	0.43 - 13.0	Not identified
K-Means-4	No	Elbow Method (Inertia)	0.38 - 41.4	5
K-Means-4	Yes	Elbow Method (Inertia)	0.32 - 10.1	17, 18
DBSCAN-2[a]	No	Max Silhouette score	0.47 - none	2, 4, 17
DBSCAN-2[b]	Yes	Max Silhouette score	0.15 none	2, 3, 4, 13, 17, 18
DBSCAN-4[a]	No	Elbow Method (Silhouette)	0.33 - none	5, 7, 17, 18, 26
DBSCAN-4[c]	Yes	Elbow Method (Silhouette)	0.16 - none	2, 4, 13, 17, 33
FC-Means-2	No	Max Silhouette score	0.37 - 51.6	23, 33, 34
FC-Means-2	Yes	Max Silhouette score	0.43 - 13.0	Not identified
FC-Means-4	No	Elbow Method (Inertia)	0.31 - 39.2	Not identified
FC-Means-4	Yes	Elbow Method (Inertia)	0.32 - 10.1	17, 18

[a]Result of eps=19 and min_samp=10.

[b]Result of eps=18 and min_samp=7.

[c]Result of eps=10 and min_samp=3.

Table 3: Summary of the total occurrences of machines of interest identified by the various models.

ID MACHINES	17	18	2	4	5	33	13	7	3	34	23	26
OCCURRENCES	6	4	3	3	3	2	2	1	1	1	1	1

4.2 Discussion

For each cluster found, it is necessary to map the inherent characteristics that isolate one from the others. The implications of each group and latent opportunities in this analysis are under continuous discussion with the stakeholders. In general, it is expected that most clusters are associated with groupings of only warnings and a mix of warnings and alarms events, both cases in proportions that are common to the process. Minority classes are the ones likely to have anomalous events that must be analyzed with greater care.

Following Lima et al.[12], The Bayesian network analysis identified that there may be a failure in the process regarding the quality of energy in the weld on the machine

id=5. The root cause is believed to be associated with a failure in the milling process of the welding electrodes, or in the cables and measurement and power circuits of this machine. There is also the possibility that the welding controller is not able to supply the power and energy required for the application of the spot, either due to a wear event (gaps, contact faults), mechanical conditions of alignment and orthogonality or due to electrical defects in electronic components of the welding controller itself.

The machines shown in Table 3 gave rise to greater opportunities for maintenance intervention. Some were not even in the radar of prioritization and these results turned out to be of great importance. A great emphasis was given to the machines id=17 and id=18, as they appear in several models as machines of interest. The machines were inspected on the shop floor and issues such as the early abrasion of the welding electrodes were raised and treated, restoring their base conditions.

5 Conclusions

5.1 Conclusion

One of the main goals of this work was to explore the data in search of anomalies that could lead to latent opportunities for the priorization of activities in specific machines. Data was migrated from the on-premises SQL database to the cloud, where it was consumed for processing and analysis. A data preprocessing was carried out in order to prepare and model them to obtain the necessary information for the purpose of finding anomalous patterns in the data.

The types of alarms are very unbalanced due to the normal operation of the welding process. Most of the data is composed of Warnings, which do not necessarily imply losses in the production process. Some warnings may simply mean records of the normal functioning of the process, such as records of milled electrodes or signaling that these need to be milled, although in some cases, prealarms, they may be indicative of errors that may occur later.

From the results obtained with the work, the authors came with the definition of two methodologies to identify machines of interest. The first one consists of scanning all types of machine failures still in the data exploration stage. When finding a machine with a variance above the others, it is considered a machine of interest for inspections and close attention of the Maintenance team. The second methodology, associated with the result of the Unsupervised Clustering algorithm, consists of classifying the events organized by day and machine, in which the database columns are composed of each possible warning or error in the process, with the values being the amount of event occurrences for the machine-data pair. It was possible to apply several different models to this data and identify the records grouped into minority classes as machines of interest.

5.2 Future Work

Other datasets, such as the spot welding process parameters and measurements *ExtMeasuresProt_V*, are likely to be used in the future to improve the findings produced with this work, to improve analysis and associate failure modes in more detail. Also, the use of this dataset might help to identify new priorities that also benefit quality control, not exclusively the maintainability of the welding process.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809917307130

[2] A. Parida and U. Kumar, "Maintenance productivity and performance measurement," in *Handbook of maintenance management and engineering*. Springer, 2009, pp. 17–41.

[3] B. Zhou, Y. Svetashova, S. Byeon, T. Pychynski, R. Mikut, and E. Kharlamov, "Predicting quality of automated welding with machine learning and semantics: a bosch case study," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2933–2940.

[4] H. Zou, "Clustering algorithm and its application in data mining," *Wireless Personal Communications*, vol. 110, no. 1, pp. 21–30, 2020.

[5] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE, 2014, pp. 232–238.

[6] J. Nayak, B. Naik, and H. Behera, "Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014," *Computational intelligence in data mining-volume 2*, pp. 133–149, 2015.

[7] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. CRC press, 2010.

[8] J. Pearl, "Bayesian networks," 2011.

[9] M. Subramaniyan, A. Skoogh, A. S. Muhammad, J. Bokrantz, B. Johansson, and C. Roser, "A data-driven approach to diagnosing throughput bottlenecks from a maintenance perspective," *Computers & Industrial Engineering*, vol. 150, p. 106851, 2020.

[10] C. Roser, M. Nakano, and M. Tanaka, "Comparison of bottleneck detection methods for agv systems," in *Winter Simulation Conference*, vol. 2, 2003, pp. 1192–1198.

[11] C. Yu and A. Matta, "A statistical framework of data-driven bottleneck identification in manufacturing systems," *International Journal of Production Research*, vol. 54, no. 21, pp. 6317–6332, 2016.

[12] E. Lima, E. Gorski, E. F. Loures, E. A. P. Santos, and F. Deschamps, "Applying machine learning to ahp multicriteria decision making method to assets prioritization in the context of industrial maintenance 4.0," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2152–2157, 2019.

[13] Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, and E. Kharlamov, "Ontology-enhanced machine learning: a bosch use case of welding quality monitoring," in *International Semantic Web Conference*. Springer, 2020, pp. 531–550.

The Maintenance of Top Frameworks and Libraries Hosted on GitHub: An Empirical Study

Yi Huang, Xinjun Mao, Zhang Zhang
National University of Defense Technology, Changsha, China
{huangyi20, xjmao, zhangzhang14}@nudt.edu.cn

***Abstract*—The number of repositories on GitHub is huge and growing rapidly. However, most repositories are inactive, while active maintenance is essential in choosing a project. In this paper, we study the maintenance of top (i.e., most starred) frameworks and libraries hosted on GitHub, for they can be widely reused and in critical positions in the dependency network, so their maintenance status is significant. Furthermore, their maintenance practices may inspire other projects to thrive on the collaborative development platform. By investigating their adoption of recommended Open Source Software (OSS) maintenance practices and recent maintenance activities, and the association between maintenance status and usage, we find that: (1) more than 20% of the top frameworks and libraries have no commit for more than one year; (2) Some maintenance practices (e.g., codes of conduct) have relatively low adoption rates, while continuous integration has a high adoption rate of around 80%; (3) the maintenance status may have an effect on the usage frequency.**

***Index Terms*—software maintenance, open source guide, software dependency**

I. Introduction

Software maintenance is critical and lasts the longest in the software life cycle. Also, it is an essential factor for developers to consider when choosing software [1]. Over the years, with the rise of open source and the emergence of code hosting platforms, more and more companies and developers maintain their projects on code hosting platforms, especially on GitHub. As the world's largest code hosting platform, GitHub has hundreds of millions of repositories and tens of millions of users, and the number is still proliferating. However, according to research [2], though the total number of repositories is vast, most of the repositories hosted on GitHub are inactive. The finding inspired us to think: are the top (e.g., most starred) projects hosted on GitHub active? Furthermore, how well are they maintained? Especially the top frameworks and libraries—while they are just the tip of the iceberg, they can be critical—they rank high, attract considerable attention, and are more likely to be reused by other projects to save development costs and increase efficiency. Therefore, their maintenance status can significantly affect related projects. Additionally, their practical experience in development and maintenance may inspire and guide other projects to thrive on GitHub.

This paper conducts an empirical study of the top frameworks and libraries in GitHub to investigate their adoption of recommended OSS maintenance practices, recent maintenance activities, and the association between maintenance status and usage. We believe that on the collaborative development platform GitHub, the maintenance complexity is greater than the maintenance complexity of individual or single team development. Attracting and retaining contributors and the interaction between developers are critical. Besides, the interaction between developers is not limited to maintaining code but also includes documentation maintenance, issue resolution, and project discussion. These aspects reflect the maintenance of the project and are worthy of in-depth exploration. In this paper, we address the following three research questions:

RQ1. (Maintenance Practices Adoption) How well do the top frameworks and libraries follow the recommended OSS maintenance practices? The purpose of the question is to investigate the adoption of some recommended maintenance practices by top frameworks and libraries. These practices are important factors for contributors to decide whether to contribute to the project [3].

RQ2. (Maintenance Activity) What is the level of recent activity present in the top frameworks and libraries? While these frameworks and libraries received many stars, they may be dormant, decayed, or even deprecated. Once developers use the deprecated libraries or frameworks, this can be a potential risk to their projects. Therefore, it is necessary to investigate their recent maintenance activities.

RQ3. (Usage) How often are the top frameworks and libraries used by other projects in GitHub? The goal of the question is to investigate the usage of top frameworks and libraries, and the association between maintenance status and usage.

Paper Organization. The rest of the paper is organized as follows. The information about our dataset is present in Section II. Our research methods and results are presented in Section III for RQ1, Section IV for RQ2, and Section V for RQ3. In Section VI, we provide an overview of related work. Section VII concludes our work.

II. Dataset

First, we referred to the previous work [4] and took 5,000 as the threshold to select the top-5,000 most starred repositories hosted on GitHub (In November 2021). Then referring to the classification criteria and results of the existing paper [5], we

DOI reference number: 10.18293/SEKE2022-092

manually selected frameworks and libraries repositories from the top-5,000 repositories, and we used the following strategies to select them: first, we selected the repositories that have keywords such as *framework* and *library* in their descriptions or README files; second, we selected the repositories that can be found in package managers (e.g., Maven, PyPI); after the above procedures, we then manually checked the repositories to assure the correctness of the selection.

Finally, we got 2,092 repositories. We collected the basic information (e.g., the number of stars, forks, commits, contributors). Besides, we collected events of issues, pulls, and commits for the repositories for the recent year between November 2020 and October 2021 through GitHub REST API. To investigate the usage of top frameworks and libraries by other repositories hosted on GitHub, we further selected repositories with more than 500 stars and had at least one commit in the past three months (for quality assurance), resulting in an initial set of 24,605 repositories. Then we selected the repositories with manifest files from the initial set, and we ended up with 14,666 repositories as potential client projects. We used libraries.io[1] to gather their dependency information. The more detailed data for analysis and its extraction process will be elaborated in the following *methodology* subsections. The data and scripts for the replication of this study are available[2].

TABLE I: Data Basic Statistics

Metric	Min	25%	Median	75%	Max	Average
Star	5,319	6,667	8,955	13,501	189,629	12,593
Fork	2	712	1,291	2,290	85,709	2,229
Age (weeks)	12	276	372	475	717	376
Commit	3	449	1,158	2,862	121,196	3,292
Contributor	1	34	85	181	4,377	163

Table I shows statistics on the number of stars, number of forks, age, number of commits, and number of contributors for the repositories in our dataset.

III. RQ1: Maintenance Practices Adoption

A. Methodology

TABLE II: Studied Maintenance Practices

Dimension	Practices	Rationale
Documentation	contribution guidelines	To guide contributors do good work
	codes of conduct	To define community standards to facilitate healthy community behavior
	template	To provide guidance for opening issues or pull requests
	good first issues	To highlight opportunities for people to contribute
	homepage	To show all kinds of information related to the project
Community	discussion	To provide a collaborative communication forum to talk about the project
	chat platform	Similar to the discussion
Automation	continuous integration	To improve the productivity of the project team

[1]https://libraries.io/

[2]10.6084/m9.figshare.18665744

Table II shows the recommended OSS maintenance practices studied in our paper. Some practices are recommended by GitHub, and some are recommended by previous work [6], [3], [4]. We divided them into three dimensions: *documentation*, *community* and *automation*.

Documentation Dimension. In this dimension, we considered five practices: adoption of *contribution guidelines*, *codes of conduct*, *template*, *good first issues* and *homepage*. These are practices recommended by GitHub to set projects for healthy contributions. To investigate their adoption, we searched for the relevant files (e.g., CONTRIBUTING.md, CODE_OF_CONDUCT.md, ISSUE_TEMPLATE.md) from the repositories' directories and checked the issue labels.

Community Dimension. In this dimension, we considered the usage of *discussion* and *chat platform*. The *discussion* refers to the new feature—GitHub Discussions. As for the usage of *chat platform*, we investigated whether the frameworks and libraries repositories have adopted Slack, Discord, or Gitter.

Automation Dimension. In this dimension, we considered the usage of *continuous integration*, and we investigated whether the top frameworks and libraries repositories have adopted Travis CI, GitHub Actions, or CircleCI.

First, we classified the collected repositories into five groups: the *top* group, which refers to the collection of top-500 repositories by the number of stars; the *bottom* group; the *active* group, which refers to the collection of repositories with at least one commit in the past month (the count is 1,034, 49.43%); the *inactive* group, which refers to the collection of repositories with no commits in the past year (the count is 448, 21.24%); the *all* group, which refers to the collection of all repositories in our dataset. The purpose of this grouping is to present the statistical differences between the top and bottom repositories, active and less active repositories, and the overall statistical characteristics of all collected repositories. Then, according to the classification of the groups, we calculated and analyzed the corresponding adoption rates of the above practices. The grouping shows that although these top frameworks and libraries received many stars and ranked high, more than 20% of them have not committed for more than one year.

B. Results

Table III shows the adoption rate of each practice by each group. Regardless of the group, the most followed practices is *continuous integration*. For *continuous integration*, the *inactive* group has the lowest adoption rate at 58.48%, while the *active* group has the highest adoption rate at 88.88%. Nevertheless, the adoption rates of *codes of conduct*, *good first issues*, *discussion* and *chat platform* are relatively low, not exceeding 50% in each group. Moreover, for each practice, the *inactive* group has the lowest adoption rate; and the adoption rates of *codes of conduct*, *template*, *good first issues* and *discussion* in *inactive* group are significantly low, which are 1/6, 1/4, 1/4 and 1/34 of the highest adoption rates.

Figure 1 shows the frequency distribution of the number of practices adopted by each group, where the *y*-axis repre-

TABLE III: Percentage of Repositories Following Recommended Maintenance Practices

Practice	Active	Inactive	Top	Bottom	All
contribution guidelines	65.57%	31.70%	71.20%	39.60%	53.11%
codes of conduct	34.43%	6.03%	36.00%	16.60%	23.57%
template	71.18%	17.19%	73.00%	37.60%	51.77%
good first issues	36.94%	9.15%	32.60%	23.80%	26.86%
homepage	76.11%	54.24%	80.60%	58.00%	67.97%
discussion	40.14%	1.79%	36.40%	17.20%	24.81%
chat platform	38.39%	15.83%	40.60%	20.40%	29.78%
continuous integration	88.88%	58.48%	88.40%	73.20%	79.73%

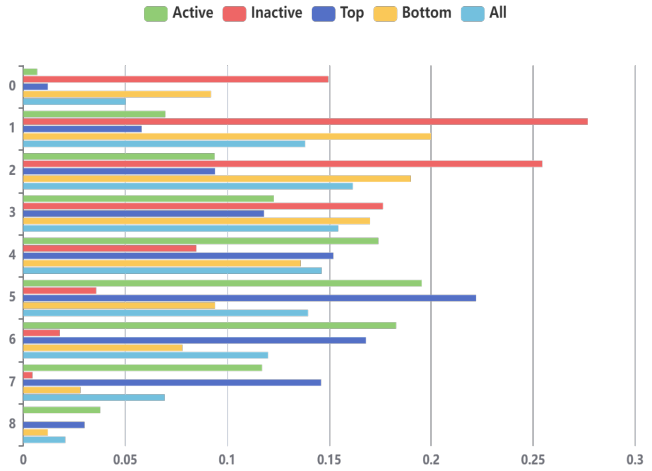


Fig. 1: Frequency Distribution of The Number of Practices Adopted by Different Groups

sents the number of practices and the *x*-axis represents the frequency. For the *inactive* group, the number of practices adopted mainly (84%) falls into the range 0 to 3. For the *active* and *top* groups, the top-3 number of practices adopted are in the range of 4 to 6. Few frameworks and libraries have adopted all the above practices though they are in active status. For *active* group, only 39 (3.77%) of the 1,034 repositories have adopted all eight studied practices.

IV. RQ2: MAINTENANCE ACTIVITY

A. Methodology

TABLE IV: The Metrics of Maintenance Activities

Dimension	Metric	Description
Commit	commit_count	The number of commits
Release	release_count	The number of releases
Issue	issue_count	The number of issues
	issue_closed_average_time	The average time of closing an issue
	issue_closed_ratio	The ratio of closed issues
	issue_replied_ratio	The ratio of replied issues
	issue_replied_average_time	The average time of replying to an issue
Pull Request	pull_count	The number of pull requests
	pull_closed_ratio	The ratio of closed pull requests
	pull_closed_average_time	The average time of closing a pull request
	pull_merged_ratio	The ratio of merged pull requests
	pull_merged_average_time	The average time of merging a pull request
	pull_replied_ratio	The ratio of replied pull requests
	pull_replied_average_time	The average time of replying to a pull request

As shown in Table IV, we investigated the maintenance activities of the top frameworks and libraries repositories in the past year (from November 2020 to October 2021) from four dimensions: *commit*, *release*, *issue*, and *pull request*. And we used the groups introduced in RQ1.

B. Results

TABLE V: Maintenance Activities of *active*, *inactive*, *top*, *bottom*, and *all* Groups.

Metric	Active	Inactive	Top	Bottom	All
commit_count	475	0	416	181	248
release_count	25	0	17	6	13
issue_count	314	14	376	91	178
issue_closed_ratio	63.92%	15.20%	58.09%	41.50%	47.62%
issue_closed_average_time	172.34h	257.65h	165.40h	218.91h	215.23h
issue_replied_ratio	76.58%	31.95%	72.48%	57.83%	62.97%
issue_replied_average_time	48.10h	472.99h	84.29h	245.95h	180.07h
pull_count	346	4	389	100	183
pull_closed_ratio	89.29%	11.37%	74.53%	57.80%	64.78%
pull_closed_average_time	108.38h	412.90h	165.24h	228.00h	246.49h
pull_merged_ratio	66.72%	0.09%	50.17%	39.47%	44.55%
pull_merged_average_time	100.04h	183.55h	137.26h	205.19h	171.50h
pull_replied_ratio	60.09%	15.67%	56.63%	41.00%	47.26%
pull_replied_average_time	69.76h	508.34h	159.89h	216.09h	234.37h

The results in Table V show that the *active* group outperforms the other groups in most metrics while the *inactive* group lags behind the other groups in most metrics. For the average number of commits and releases, the *active* group has the maximum values, 475 and 25, respectively, while the *inactive* group has the minimum values, both zero. For the average number of issues and pull requests, the *top* group has the maximum values of 376 and 389, while the *inactive* group has the minimum values of 14 and 4, respectively. Besides, the *active* group has the highest ratios of closed issues, replied issues, closed pull requests, merged pull requests, and replied pull requests, which are 63.92%, 76.58%, 89.29%, 66.72%, and 60.09%, respectively, while the *inactive* group has the lowest which are 15.20%, 31.95%, 11.37%, 0.09%, and 15.67% respectively.

Interestingly, we found that: in *inactive* group, the average time of replying to an issue is longer than the average time of closing an issue; the average time of replying to a pull request is longer than the average time of closing or merging a pull request; however, the above situation is reversed in *active* group.

V. RQ3: USAGE

A. Methodology

In this section, as for the usage of top frameworks and libraries, we investigate their usage frequency and outdated usage. We used libraries.io to extract the dependencies of each potential client project. The libraries.io has organized all the manifest files (e.g., pom.xml, package.json) for the project. Finally, we extracted 216,504 dependencies.

B. Results

TABLE VI: Statistics on the usage frequency and outdated usage ratio

Metric	Min	25%	Median	75%	Max	Average
Usage Frequency	0	0	4	33	13,069	102
Outdated Usage Ratio	0	9.22%	49.77%	80.00%	100.00%	47.10%

Table VI shows statistics on the usage frequency and outdated usage ratio. We define the outdated usage ratio as the number of outdated uses of a framework or library divided by its total uses. Of the 2,092 studied top frameworks and libraries, 1,362 are used at least once, only 285 do not have outdated usage, and 146 projects are wholly used with outdated versions.

We applied the Mann-Whitney U test to analyze the statistical significance of the difference between the top-100 most used frameworks and libraries and the unused frameworks and libraries in the metrics mentioned in RQ2, and we used Cliff's delta [7] to show the effect size of the difference. We found a statistically significant difference between them in all metrics, and all 14 metrics have large effect sizes. Further, We calculated the median and the mean of the 14 metrics of the top-100 most used and unused projects and found that the former outperforms the latter in all metrics. For example, the median number of commits in the past year of the former is 74 while the latter is 12, and the issue closed ratio of the former is 70.14% while the latter is 35.15%. Therefore, the maintenance status may have an effect on the usage frequency. Then, we used the same methods above to analyze the projects that do not have outdated usage and projects that are wholly used with outdated versions. We found a statistically significant difference between them in all metrics, and 10 of 14 metrics have large effect sizes, while the other metrics have medium effect sizes. Further, we found that the latter outperforms the former in 13 metrics (except the issue closed average time).

VI. Related Work

Much work has been done studying software maintenance. There is some work [8], [9] focused on investigating or measuring the maintenance status of projects; some work [10] focused on the barriers faced by contributors (e.g., peripheral contributors); some work [4], [6], [11], [3] focused on the recommended OSS maintenance practices that may guide or automate the maintenance and contribution process. Coelho et al. [8] proposed a machine learning model to identify unmaintained GitHub projects and defined a metric to measure the level of maintenance activity of GitHub projects. Lee et al. [10] conducted an online survey to investigate the barriers one-time code contributors faced when contributing to FLOSS projects and highlighted the significance of timely feedback and guidance through the patch submission process. Hilton et al. [6] studied the usage, costs, and benefits of continuous integration in open source projects. They found that the overall percentage of projects using CI continues to grow, and CI helps projects release more frequently and accept pull requests faster. Alderliesten et al. [11] initially explored the "good first issues" label and found that though they are effective at developer onboarding and considered useful, their types need to be refined to match the types of initial contributions.

VII. Conclusion

In this paper, we conducted an empirical study to investigate the maintenance of top frameworks and libraries hosted on GitHub. We found that some OSS recommended maintenance practices are not widely adopted even in the top frameworks and libraries. For example, the adoption rates of codes of conduct and good first issues are 23.57% and 26.86%. Further, we used quantity, proportion, and response time as metrics to analyze the recent maintenance activities of the top frameworks and libraries. Moreover, we found that the maintenance status may have an effect on the usage frequency. In future work, we plan to propose a unified measure of open source project maintenance status.

References

[1] E. Larios Vargas, M. Aniche, C. Treude, M. Bruntink, and G. Gousios, *Selecting Third-Party Libraries: The Practitioners' Perspective*. New York, NY, USA: Association for Computing Machinery, 2020, p. 245–256. [Online]. Available: https://doi.org/10.1145/3368089.3409711

[2] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. German, and D. Damian, "The promises and perils of mining github," 06 2014.

[3] D. Sholler, I. Steinmacher, D. Ford Robinson, M. Averick, M. Hoye, and G. Wilson, "Ten simple rules for helping newcomers become contributors to open projects," *PLoS Computational Biology*, vol. 15, September 2019.

[4] J. Coelho and M. Valente, "Why modern open source projects fail," 08 2017, pp. 186–196.

[5] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 334–344.

[6] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 426–437.

[7] G. Macbeth, E. Razumiejczyk, and R. D. Ledesma, "Cliff's delta calculator: A non-parametric effect size program for two groups of observations," *Universitas Psychologica*, vol. 10, no. 2, pp. 545–555, 2011.

[8] J. Coelho, M. Valente, L. Milen, and L. Silva, "Is this github project maintained? measuring the level of maintenance activity of open-source projects," *Information and Software Technology*, 02 2020.

[9] G. Avelino, E. Constantinou, M. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," 09 2019, pp. 1–12.

[10] A. Lee, J. C. Carver, and A. Bosu, "Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: A survey," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, pp. 187–197.

[11] J. W. D. Alderliesten and A. Zaidman, "An initial exploration of the "good first issue" label for newcomer developers," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2021.

Evaluating the Sustainability of Computational Science and Engineering Software: Empirical Observations

James M. Willenbring
Software Engineering & Research Department
Sandia National Laboratories*
Albuquerque, New Mexico
jmwille@sandia.gov

Gursimran Singh Walia
School of Computer and Cyber Sciences
Augusta University
Augusta, Georgia
gwalia@augusta.edu

***Abstract*— Software sustainability is critical for Computational Science and Engineering (CSE) software. It is also challenging due to factors ranging from funding models to the typical lifecycle of a research code to the inherent challenges of running fast on the newest architectures. Furthermore, measuring sustainability is challenging because sustainability consists of many complex attributes. To identify useful metrics for measuring CSE software sustainability, we gathered data from multiple freely available sources, including GitHub, SLOCCount, and Metrix++. This paper discusses the challenges practitioners face when measuring the sustainability of CSE software. We present an analysis of data with associated observations and future directions to better understand CSE software sustainability and how this work can be used to support decisions and improve sustainability by observing trends in metrics over time.**

I. INTRODUCTION

Software sustainability is a key issue in the Computational Science and Engineering (CSE) domain. CSE software projects often begin as a research activity. Software engineering concerns are secondary to the research objectives driving the project [4]. While some research activities fail (as is consistent with research), successful projects often result in software with a very long useful life. So while there is a risk that investing early on in sustainability may prove to be a wasted effort in a sense if the research activity fails, the penalty for not investing in sustainability for initially successful projects is high. Projects developed without sustainability in mind eventually become fragile. For example, poor designs limit extensibility and evolvability, and insufficient testing leads to a lack of maintainability. Other factors (e.g., funding models, developer training, staffing challenges, etc.) can also contribute to a lack of sustainability [2].

Research and practice indicate that sustaining CSE software is inherently complex, as discussed herewith. CSE software utilizes cutting-edge algorithms and language features to promote performance on the world's largest supercomputers, particularly on new architectures. Codes are often significant - hundreds of thousands or millions of code lines, and commonly use several third-party software packages (many of which are open source). While these challenges are not new, they have grown in recent years. One of the author's previous work offered some simple ways to improve the quality of CSE software [4]. Our suggestions included the use of source code management, issue tracking, documented processes, source-centric documentation, pair programming, continuous process improvement, and other software practices. While the principles remain relevant, developers' environment has evolved to be much more complex.

Prior work lists several sustainability attributes: extensibility, interoperability, maintainability, portability, reusability, scalability, and usability [10]. Motivated by an array of literature on sustainability attributes, this work analyzes the most critical aspects of software sustainability in the context of CSE software. Several groups have gathered, analyzed, and/or defined metrics with a focus towards software sustainability or an attribute of software sustainability [15] [7] [5] [9] [8] [1]. However, previous efforts have not closely examined metrics in the context of CSE software sustainability. Furthermore, prior literature lacks tool support for researchers in gathering more advanced metrics, such as those in [7] for highly complicated (C++) codebases.

This research aims to charazterize better and identify barriers to CSE software sustainability, as well as reduce those barriers by identifying tools and techniques to support decision making focused on improving software quality.

II. APPROACH

We analyzed CSE software projects openly available on GitHub, GitLab, and Bitbucket. Although not all of the repositories go back to the beginning of the projects (some first used another version control system and snapshotted the code into git), all of the repositories have substantial history to examine. Additionally, due to scientific software community involvement, we have access to the contributors to many of the projects being analyzed.

Our strategy for measuring sustainability focuses on the various component attributes before embarking on a more

*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the US Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.
This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND2021-7711 C.

DOI reference number: 10.18293/SEKE2022-154.

holistic analysis. An initial broad set of representative CSE software projects was carefully selected for the study. We started by considering software that comprises the Extreme-scale Scientific Software Development Kit (xSDK). The xSDK [14] project was created to improve the interoperability and sustainability of scientific libraries that are common dependencies for scientific software.

Next, we chose packages that represented both large and small source code bases and development teams and a variety of primary development institutions (four US national laboratories and two universities). We also wanted projects with lengthy histories, which would allow us to go back and look at changes over time. Six out of the seven projects chosen are also currently funded in part under Math Libraries within the Exascale Computing Project's (ECP) Software Technologies (ST) thrust [3]. A group of experts initially chose this software group to be part of ECP based on each software package's current and potential value to high-performance computing. Further, ECP ST is very interested in improving the sustainability of this software. Below is a brief description of the seven software projects chosen for our metric collection activity. The names of the projects have been changed to guard against unintended conclusions being drawn about the sustainability of any specific project.

Project 1 includes linear and non-linear solvers as well as preconditioners for partial differential equation-based systems of equations. **Project 2** is a collection of solvers and enabling technologies used for large-scale, complex multi-physics engineering and scientific problems. **Project 3** is a distributed memory direct LU solver for non-symmetric sparse linear systems of equations. **Project 4** provides algebraic multigrid sparse preconditioners and solvers. **Project 5** is a dense linear algebra library that provides linear, least squares, eigen, and S.V.D. solvers. **Project 6** is a finite element and adaptive mesh refinement code. **Project 7** is a sparse linear and nonlinear eigenvalue solver package.

TABLE 1: Language, SLOC, Contributors & Commits

Package	Language	SLOC	Contribs	Commits
Project 1	C 83%	796123	198	82663
Project 2	C++ 82%	4179781	250	95384
Project 3	C 96%	80752	14	645
Project 4	C 81%	441057	37	11587
Project 5	C++ 45%	317082	51	8086
Project 6	C++ 100%	293062	114	14668
Project 7	C 91%	109559	26	9045

The purpose of gathering these metrics is to analyze the correlation of the metrics with aspects of sustainability. Because no single metric fully reflects the sustainability of a software project, we look at several metrics, including some directly related to the source code, such as complexity and lines of code, and others that are not directly related to the code itself, such as a number of commits and contributors. Additionally, we also analyzed metric trends over time. Researchers and practitioners can utilize trends to better understand if a codebase is becoming more or less sustainable (e.g. observing an increasing or decreasing number of contributors, cyclomatic complexity, or maintenance index).

III. DATA ANALYSIS AND RESULTS

Our approach involved gathering metrics from accessible sources and tools. The first set of results consists of metrics gathered from development snapshots of the seven codes we chose from May 2021. The second set of results includes metrics collected from five snapshots in time for each of the seven codes from May 2017-2021 (once per year). We describe each of these sets of metrics below.

The first set of results obtained information about the number of contributors and the number of commits from GitHub and Gitlab, and from the git command line (as Bitbucket does not supply this information). We also gathered metrics involving the use of the tool SLOCCount [11] to obtain the primary programming language for each project, along with the percentage of lines in the project of the primary language and the total number of source lines of code (SLOC).

The second set of metrics was collected using Metrix++ version 1.7.0 [6]. To capture yearly snapshots, git commands of the form *git checkout `git rev-list -n 1 --first-parent --before="2019-05-24 00:00" <primary_branch_name>* were used. The metrics included in this set are:

- *Maximum Complexity*: The maximum cyclomatic complexity found in the code.
- *Average Complexity*: The average cyclomatic complexity found in all regions of the code.
- *Lines of code*: The total number of lines of code. Note Metrix++ considers only C, C++, and Java code (not Fortran, Python, or scripts).
- *Maintenance Index:* A measure of maintainability computed from cyclomatic complexity and lines of code. A lower value indicates a higher level of maintainability.

TABLE 2: Max Cyclomatic Compexity

Package	2017	2018	2019	2020	2021
Project 1	1786	1788	2319	540	540
Project 2	648	648	648	547	547
Project 3	202	204	301	301	301
Project 4	649	765	815	877	913
Project 5	261	261	261	261	261
Project 6	114	114	137	134	238
Project 7	86	86	86	86	83

TABLE 3: Average Cyclomatic Compexity

Package	2017	2018	2019	2020	2021
Project 1	4.63	4.69	4.88	4.69	4.58
Project 2	2.59	2.47	2.46	2.39	2.30
Project 3	12.50	11.97	10.62	10.31	10.36
Project 4	7.14	6.65	6.50	6.48	6.33
Project 5	5.45	5.40	5.27	5.30	5.33
Project 6	2.34	2.33	2.14	2.48	2.49
Project 7	3.95	3.97	4.14	4.11	4.10

TABLE 4: Lines of Code (in 1,000's)

Package	2017	2018	2019	2020	2021
Project 1	483	525	584	613	657
Project 2	2682	3082	3076	3279	3414
Project 3	55	58	79	81	86
Project 4	410	359	365	390	405
Project 5	128	131	136	137	138
Project 6	107	122	154	216	284
Project 7	71	79	84	89	95

TABLE 5: Maintenance index computed by Metrix++ using complexity and lines of code data

Package	2017	2018	2019	2020	2021
Project 1	1.40	1.41	1.43	1.42	1.41
Project 2	1.21	1.18	1.18	1.19	1.18
Project 3	2.35	2.29	2.08	2.05	2.08
Project 4	1.81	1.78	1.77	1.79	1.79
Project 5	1.76	1.71	1.70	1.70	1.71
Project 6	1.21	1.21	1.19	1.24	1.24
Project 7	1.31	1.32	1.33	1.33	1.33

IV. DISCUSSION OF RESULTS

We discuss significant results concerning metrics reported in the previous section focused around key themes and contributions to understanding CSE software sustainability.

SLOC and Contributors: - Poor software design, for example, poor understandability, has a greater than linear impact as SLOC increases in terms of maintainability and evolvability. Code size metrics such as SLOC can be useful to measure over time. For example, SLOC growth in excess of feature set growth may indicate the need to refactor.

A very low number of contributors can be a sustainability risk in that the knowledge of the code is owned by a small group of people. The number of contributors to the seven codes varies by more than an order of magnitude. Further, the ratio of SLOC to contributors may speak to maintainability. Less code per contributor means fewer lines that each contributor needs to maintain. Projects 2 and 4 have significantly higher SLOC to contributor ratios than the other five codes.

Complexity: A lower average complexity should enhance readability and maintainability, all else equal. Interestingly, the average complexity of Project 3 is nearly twice the average complexity of the next highest sample code.

TABLE 6: SLOC and Contributors

Package	SLOC	contributors	SLOC/contrib
Project 1	796123	198	4021
Project 2	4179781	250	16719
Project 3	80752	14	5768
Project 4	441057	37	11920
Project 5	317082	51	6217
Project 6	293062	114	2571
Project 7	109559	26	4214

While cyclomatic complexity does not capture all aspects of maintainability, by definition, it does reflect the number of paths through the code. If this value is growing over time, it can increase the maintenance burden. The Max complexity data gathered in Table 2 is interesting in that for three codes the value grew between 2017 and 2021, for two the value fell, and for two it remained nearly or exactly the same. Metrix++ has a "*hotspot*" feature that allows a person to identify regions with a complexity greater than a given *threshold*, which can be used to support a targeted refactoring effort.

Maintenance Index: The Maintenance index data in Table 5 does not change dramatically for any packages over time. This is again not surprising because these are large, established code bases, and in any given year, large portions of the codebase do not change. The most significant change in the codes' value comes from Project 3 between 2018 and 2019, with the Maintenance index falling from 2.29 to 2.08. We note that in the same period, Table 4 shows that the Lines of code increased substantially from 58,034 to 79,069. It is reasonable that a 36% increase in the size of the codebase would cause a noticeable decrease in the maintainability index if the new code was written more maintainably.

Similar to the previous observations for complexity, we feel that looking at changes in the maintenance index both over time and addressing maintenance "hotspots" could improve the maintainability and sustainability of codes. In addition, these checks can be automated in continuous integration or nightly processes and tracked over time to identify trends.

Metrix++ Hotspot Feature: As mentioned above, the hotspot feature in Metrix++ is a useful tool that allows regions of code to be identified that exhibit a metric value above a user-specified threshold. The tool is both simple to use and powerful. For example, the below command identified five regions of code in Project 4 with cyclomatic complexity equal to or greater than 500: *metrix++ limit --db-file=proj4.2019.lines.complex.maint.db --max-limit=std.code.complexity:cyclomatic:500*

The tool supports more advanced features that allow it to be used in an automated testing environment. The return code of the limit function is equal to the number of instances in the code where the metric threshold specified is exceeded. Therefore, automated, pre-push testing can prevent complex code from being added by checking this return code. Alternatively, automated metrics can be gathered and provided to code reviewers to use in their analysis.

While finding all areas of high complexity might be useful in some contexts, often it is preferable to consider only new or modified (the term Metrix++ uses for this option is "touched") code. For those cases, specify a previous version of a database file (using *--db-file-prev*) to compare against: *metrix++ limit --db-file=proj4.2019.lines.complex.maint.db --db-file-prev=../2018-05-24/proj4.2018.lines.complex.maint.db --max-limit=std.code.complexity:cyclomatic:500 --warn-mode=touched*

In our example, three of the five regions were touched. Finally, because refactoring all code that is touch may not be practical, Metrix++ supports a "*trend*" feature (using *--warn-mode=trend*, rather than *touched*) that only identifies code

for which the metric in question has gotten worse since the previous state. In our example, two of the three touched regions exhibited a negative trend.

In summary, the hotspot feature in Metrix++ allows a team to not only identify regions of their code of concern (for example, due to high complexity or maintenance index), but also allows the team to automate the tracking and even prevention of additional regions of concern. This feature can be used to support maintainability and sustainability.

V. CONCLUSION AND RELEVANCE TO INDUSTRY

While it is never appropriate to make broad conclusions based on a single metric, the metrics we studied provide quantitative data that can be used to support decisions. We caution, for example, against using any simple metrics to claim that one code is more sustainable than another. That said, a developer performing a code review can benefit from using the Metrix++ hotspot feature by considering changes in maintenance index or max or average complexity metrics in the broader context of the proposed changes.

By sampling multiple metrics for a single code base over time, practitioners can glean whether it is becoming more or less sustainable. Such information may help assess the effectiveness of changing development practices or tools. For example, a decreasing maintenance index for a code base following the adoption of a new development practice supports the hypothesis that the new practice is beneficial.

A similar approach could be used in evaluating the impact of a refactoring effort. For example, before refactoring a large chunk of code, one might gather average and maximum complexity as well as lines of code and maintenance index metrics. Then, the areas of complexity higher than a given threshold could be identified as candidates for refactoring. After the refactoring step, the metrics can be taken again to help quantify the impact of the refactoring, and metrics could be sampled periodically to understand the effects of development practices on the sustainability of the codebase.

In the future, we plan to explore sustainability factors not addressed by source code metrics, such as sustainability of dependencies, and how the sustainability of CSE software is impacted by the sustainability of the CSE software ecosystem as a whole [12]. For example, consider how common interfaces could improve software sustainability.

Another area of future work would be to consider smaller logical subsets of codebases and "hotspots" identified using Metrix++. We could also study other metrics such as contributors at a more granular level. Specifically, we could compare the number of frequent and recent contributors and total contributors to functionality in the code that is effectively orphaned (no currently assigned developers) to functionality that is more actively supported.

In an effort to help code teams and project leadership gather and effectively utilize metrics and related tools in their scientific software development efforts, we are also forming a Software Development Kit (SDK) community in the area of Tools for Code Mining and Data Analysis [13].

This research effort can make significant contributions to the understanding of the sustainability of relevant components of the CSE software stack. We analyzed the seven code projects part of the xSDK, six of which are part of the US DOE Exascale Computing Project. This allows us to base our findings on industrial representative CSE software, increasing the generalizability of our results.

Perhaps most importantly, a better understanding of software sustainability can help to identify how to design from the onset for sustainability, which has the potential to save significant developer time (and money) and prevent a lot of frustration dealing with unsustainable code.

Bibliography

[1] Bouwers, E., van Deursen, A., & Visser, J. (2013). Evaluating Usefulness of Software Metrics: An Industrial Experience Report. *Proc. Int'l Conf. Software Eng. (ICSE 13), IEEE*, 921-930.

[2] Heroux, M. A., & Allen, G. (2016, Sept). Computational Science and Engineering Software Sustainability and Productivity (CSESSP) Challenges Workshop Report. *Networking and Information Technology Research and Development (NITRD) Program*.

[3] Heroux, M. A., Carter, J., Thakur, R., McInnes, L., Ahrens, J., Munson, T., & Neeley, J. R. (2020, February 1). ECP Software Technology Capability Assessment Report. 10.2172/1606665

[4] Heroux, M. A., & Willenbring, J. M. (2009). Barely sufficient software engineering: 10 practices to improve your CSE software. *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, 15-21. 10.1109/SECSE.2009.5069157

[5] Koziolek, H. (2011). Sustainability evaluation of software architectures: A systematic review. *Proceedings of the Joint ACM SIGSOFT Conference - QoSA and ACM SIGSOFT Symposium - ISARCS on Quality of Software Architectures - QoSA and Architecting Critical Systems - ISARCS QoSA-ISARCS '11*, 3-12.

[6] *Metrix++ Web Page*. (n.d.). https://metrixplusplus.github.io/metrixplusplus/

[7] Sarkar, S., Kak, A., & Rama, G. (2008). Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software. *IEEE Transactions on Software Engineering*, *34*(5), 700-720.

[8] Sarkar, S., Rama, G. M., & Kak, A. C. (2007). API-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization. *IEEE Trans. Software Eng.*, *33*(1), 14-32.

[9] Sehestedt, S., Cheng, C.-H., & Bouwers, E. (2014). Towards quantitative metrics for architecture models. *In Proceedings of the WICSA 2014 Companion Volume (WICSA '14 Companion). ACM*, Article 5, 4 pages. http://dx.doi.org/10.1145/2578128.2578226

[10] Venters, C. C., Lau, L., Griffiths, M. K., Holmes, V., Ward, R. R., Jay, C., & J, X. (2014). The Blind Men and the Elephant: Towards an Empirical Evaluation Framework for Software Sustainability. *Journal of Open Research Software*, *2(1)*(8). http://doi.org/10.5334/jors.ao

[11] Wheeler, D. A. (n.d.). *SLOCCount*. https://dwheeler.com/sloccount/

[12] Willenbring, J. M. (2019). The Layers of CSE Software Sustainability. *2019 Collegeville Workshop on Sustainable Scientific Software (CW3S19)*. https://collegeville.github.io/CW3S19/WorkshopResources/WhitePapers/CSEswSustainabilityLayers.pdf

[13] Willenbring, J. M. & Shende S. (2022). Impacting Software Quality and Process Through the Extreme-Scale Scientific Software Stack (E4S) and Software Development Kit (SDK) Projects.

[14] *xSDK Web Page*. (n.d.). xSDK: Extreme-scale Scientific Software Development Kit. Retrieved 12 01, 2020, from http://xsdk.info

[15] Zhao, Y., Yang, Y., Lu, H., Zhou, Y., Song, Q., & Xu, B. (2015). An empirical analysis of package-modularization metrics: Implications for software fault-proneness. *Information and Software Technology*, *57*, 186-203. 10.1016/j.infsof.2014.09.006

Decisions in Continuous Integration and Delivery: An Exploratory Study

Yajing Luo[†], Peng Liang[†*], Mojtaba Shahin[‡], Zengyang Li[§], Chen Yang[¶]
[†]School of Computer Science, Wuhan University, Wuhan, China
[‡]School of Computing Technologies, RMIT University, Melbourne, Australia
[§]School of Computer Science & Hubei Provincial Key Laboratory of Artificial Intelligence and Smart Learning, Central China Normal University, Wuhan, China
[¶]School of Artificial Intelligence, Shenzhen Polytechnic, Shenzhen, China
{luoyajing, liangp}whu.edu.cn, mojtaba.shahin@rmit.edu.au, zengyangli@ccnu.edu.cn, yangchen@szpt.edu.cn

***Abstract*—In recent years, Continuous Integration (CI) and Continuous Delivery (CD) has been heatedly discussed and widely used in part or all of the software development life cycle as the practices and pipeline to deliver software products in an efficient way. There are many tools, such as Travis CI, that offer various features to support the CI/CD pipeline, but there is a lack of understanding about what decisions are frequently made in CI/CD. In this work, we explored one popular open-source project on GitHub, Budibase, to provide insights on the types of decisions made in CI/CD from a practitioners' perspective. We first explored the GitHub Trending page, conducted a pilot repository extraction, and identified the Budibase repository as the case for our study. We then crawled all the closed issues from the repository and got 1,168 closed issues. Irrelevant issues were filtered out based on certain criteria, and 370 candidate issues that contain decisions were obtained for data extraction. We analyzed the issues using a hybrid approach combining predefined types and the Constant Comparison method to get the categories of decisions. The results show that the major type of decisions in the Budibase closed issues is *Functional Requirement Decision* (67.6%), followed by *Architecture Decision* (11.1%). Our findings encourage developers to put more effort on the issues and making decisions related to CI/CD, and provide researchers with a reference of decision classification made in CI/CD.**

***Keywords*—Decision, Continuous Integration and Delivery, CI/CD, Budibase, Empirical Study**

I. Introduction

The Software Development Life Cycle (SDLC) of an information system goes through the steps of planning, creation, testing, and deployment. In the past, organizations needed to provision, operate, and maintain the build job creation, processing, and reporting themselves [1]. In recent years, DevOps, a set of practices (e.g., cloud-based continuous integration, and automated deployment) that combine software development and IT operations, enables organizations to deliver changes into production as quickly as possible, without compromising software quality [2]. **In this study**, we focus on Continuous Integration and Delivery (CI/CD), the key enabler of DevOps. CI systems automate the compilation, building, and testing of software [3]. A typical CI service is composed of three types of nodes. First, build job creation nodes queue up new build jobs when configured build events occur. Next, a set of build job processing nodes process build jobs from the queue, adding job results to another queue. Finally, build job reporting nodes process job results, updating team members of the build status [1]. CD aims at ensuring an application is always at production-ready state after successfully passing automated tests and quality checks [4].

Nowadays, there are many tools that have been available to support the CI/CD pipeline, such as Jenkins[1], Travis CI[2], Circle CI[3], and GitHub Actions[4], offering various CI/CD features. Among them, GitHub Actions is a CI/CD platform that allows developers to automate their build, test, and deployment pipeline. It was launched in 2018 and supported CI/CD from November 13, 2019. With GitHub Actions, practitioners can create workflows that build and test every pull request to their repository, or deploy merged pull requests to production.

Although many studies have been conducted in a CI/CD context [1] [5], very few of them have explored decisions made in CI/CD, how the decisions made in CI/CD differ from those made in traditional software development, and what types of decisions are made in CI/CD. We selected the Budibase repository as the case for this study due to: (1) it uses GitHub Actions to support CI/CD, (2) it is a trending repository on GitHub in the recent three months, (3) it is active in the recent three months, and (4) the number of its closed issues exceeds 1000. In short, we selected the Budibase repository as the study project and collected data from its closed issues. We then identified and extracted decisions from the dataset, and classified them using a hybrid approach.

The contributions of this paper: (1) we provide an exploratory study on the decisions made in CI/CD; (2) we classify the decisions made in CI/CD; and (3) we compare the decisions made in CI/CD with the decisions made in traditional development.

This work is supported by National Key R&D Program of China with No. 2018YFB1402800, NSFC with No. 62172311, Hubei Provincial Natural Science Foundation of China with No. 2021CFB577, and Research Foundation of Shenzhen Polytechnic with No. 6022312043K.
DOI reference number: 10.18293/SEKE2022-171

[1]https://www.jenkins.io/
[2]https://travis-ci.org/
[3]https://circleci.com/
[4]https://github.com/features/actions

The rest of the paper is organized as follows: Section II discusses related work to our study. Section III describes the research question and its rationale, the data collection, filtering, extraction, and analysis process. Section IV provides the results of the research question. Section V explains the results with their implications and comparison with the existing work. Section VI presents the threats to the validity of this study. Section VII concludes this work with further directions.

II. Related Work

A. Continuous Integration and Delivery

CI is an established software quality assurance practice and has been the focus of much prior research with a diverse range of methods and populations [6]. CD has been adopted by many software organizations to develop and deliver quality software more frequently and reliably [5].

Gallaba *et al.* [1] set out to study how CI features are being used and misused in 9,312 open source systems that use Travis CI. Widder *et al.* [6] reviewed the CI literature of 37 papers from the perspective of pain points to adoption and usage, and performed a mixed-methods conceptual replication of previously observed findings. Zhao *et al.* [2] studied the adaptation and evolution of code writing and submission, issue and pull request closing, and testing practices on hundreds of established projects on GitHub adopting Travis CI. Hilton *et al.* [3] used three complementary methods to study the usage of CI in open-source projects. As for CD, Shahin *et al.* [5] conducted a mixed-methods empirical study and presented a conceptual framework to support the process of (re-)architecting for CD.

To the best of our knowledge, there are no studies that explore the decisions made in CI/CD, how they differ from those in traditional software development, and what types of decisions are frequently made. In our study, we intended to explore the decisions made in CI/CD using an open source project (Budibase) that employs GitHub Actions.

B. Decisions in Software Engineering

Developing and maintaining a software system involves making numerous important decisions by stakeholders [7]. These decisions cover the life cycle of software development, including requirements [8], architecture design [9], project management [10], etc. [11], and each decision is typically influenced by other decisions and involves trade-offs in system properties [12]. Design decisions directly impact system quality [12], and high-quality decisions are critical to the success of projects [13]. Many researchers focus on exploring how to make high-quality and appropriate decisions that meet project objectives and maximize system benefits, analyzing the rationale behind decisions, and understanding their effects on the system.

Tang *et al.* [14] proposed a systematic approach to software design decision-making. They broke decision-making down into nine principles that can be taught, learned, and practiced, and each principle addresses one decision-making aspect that focuses on a specific type of information used in making and evaluating decisions. Shahbazian *et al.* [12] posed the problem of making complex, interacting design decisions relatively early in the project's lifecycle and outlined a search-based and simulation-based approach for helping architects make these decisions and understand their effects. Li *et al.* [11] tried to identify decisions discussed in the Hibernate developer mailing list and explore decision-making from several aspects (i.e., description, classification, underlying rationale, supporting approaches, related artifacts, and trend). Sharma *et al.* [15] explored the rationale behind 'how' the decisions are made, and made a methodological contribution by presenting a heuristics-based rationale extraction system employing multiple heuristics, and following a data-driven, bottom-up approach to infer the rationale behind specific decisions. Liu *et al.* [16] intended to understand how decisions are made in requirements engineering through the student projects in a requirements engineering course.

Different from the above works, we studied the decisions made in CI/CD by analyzing the closed issues in the Budibase repository and focusing on the types of decisions.

III. Research Design

We set the goal of this study based on the Goal-Question-Metric approach [17], which is to **analyze** the decisions in CI/CD **for the purpose of** characterizing **with respect to** the types of decisions made in CI/CD **from the point of view of** practitioners **in the context of** CI/CD of open source software. To archive this goal, we address the following key Research Question (RQ):

- **RQ**: What types of decisions are made in CI/CD?
  Rationale: Various types of decisions made in traditional software development have been discussed in literature (e.g., [11]). This RQ aims to provide a classification of the decisions made in projects using CI/CD and compare these decisions with those made in traditional software development, so that we can see the difference between CI/CD and traditional development from a decision perspective.

In the following subsections, we detail the research process (see Figure 1) used to answer this RQ.

A. Data Collection

To answer the RQ, we need to identify and collect decisions from the projects using CI/CD. GitHub is a popular and most frequently used code hosting platform for version control and collaboration. After exploring several artifacts (i.e., Github Discussions, Github Issues, Slack, Gitter, and Microsoft teams), we intended to select a suitable GitHub project that employs CI/CD practices for our study, and use GitHub Issues (developer discussions) as the data source, since closed issues include a wide range of decisions made in development. We searched for the candidate GitHub project based on the following criteria:

- **A repository running a GitHub Actions workflow**
  Given that we needed to select a project applying CI/CD. GitHub provides a CI/CD tool named GitHub Actions to

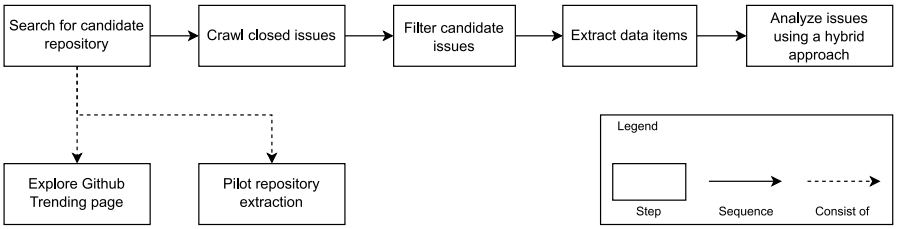


Fig. 1: Overview of the research process

allow developers to build, test, and deploy their software automatically. Github Actions provided by Github started to support CI/CD in 2019 and does not require migration from Circle CI, Jenkins, or Travis CI. We believe that there should be many projects on Github applying Github Actions in the last 2 years. Hence, we decided to look for a repository on GitHub that uses the feature of GitHub Actions.

- **A hot topic in GitHub in the last three months**
  A GitHub Trending page is available on GitHub where trending repositories are listed based on the number of times they have been starred by users every day, week, or month. This trending page provides a view of the open-source projects which the community is most excited about. We decided to trace the hot topics in GitHub in the last three months and select an appropriate repository for our study.
- **An active repository**
  We decided to choose a repository that is currently active and frequently updated (e.g., usually make changes to a file and push them to GitHub as commits, often open and merge a pull request, and have heated discussion in GitHub Issues).
- **The number of closed issues exceeds 1000**
  Since issues are the data source of this study, and the number of issues of a project is an indicator of the project size, which is an important factor to get enough data for an empirical study. We decided to look for a repository with more than 1000 closed issues.

After exploring the repositories listed on the GitHub Trending page for the past three months, we picked out several repositories (including Budibase, Rails, and Tabby), which met the above criteria. We then conducted a pilot repository selection. The first author randomly selected 50 issues from the closed issues of the candidate repositories. The purpose was to verify whether the repositories we picked out were appropriate and whether the data item (type of decision) can be extracted from the issues (see the criteria in Section III-B). Next, the first author extracted the data item from these 50 issues and then discussed the extraction results with the second author to see if the extraction results were correct. After the pilot repository selection, we finally chose Budibase[5].

Budibase is an all-in-one low-code platform for building, designing, and automating business apps with which building business apps in minutes. We crawled all the URLs of the closed issues of Budibase into an Excel sheet on January, 2022, and got 1,168 links.

[5]https://github.com/Budibase/budibase

B. Data Filtering, Extraction and Analysis

1) Filter candidate issues: The criteria for filtering issues are defined as follows:

- **If an issue contains the data item (i.e., type of decision) to be extracted, we include it.**
- **If an issue does not contain the data item to be extracted or cannot be understood, we exclude it.**

This step was conducted by the first author, and the issues that the first author could not decide were discussed among the authors to reach a consensus. After excluding the irrelevant issues, we finally got 370 issues out of the 1,168 closed issues from Budibase.

2) Extract and analyze the data item: First of all, for answering the RQ, the first author extracted the data item (type of decision) from each relevant issue. After data extraction, the first author classified the decisions extracted based on the decision types provided by Li *et al.* [11]. Then the first author discussed the classification results (i.e., by following the pre-defined types in [11]) with the second author and adjusted the classification by taking into account both the content of the extracted decisions and the labels of the issues using the Constant Comparison method [18]. We refer to this combination of the pre-defined classification and the Constant Comparison method as a hybrid approach. We obtained the final classification of decisions using the hybrid approach. We have also provided the dataset and the classification results of decisions online [19].

IV. Results

We applied the hybrid approach and classified the decisions into several types as shown in Table I, which provides the types (categories and subcategories) with their descriptions, examples, and percentages.

We got six categories of decisions made in CI/CD, namely *Requirement Decision*, *Architecture Decision*, *Management Decision*, *Build Decision*, *Testing Decision*, and *Deployment Decision*. We also got 11 subcategories of these major categories except *Architecture Decision*, which does not have any subcategories. The major type of decisions in the Budibase closed issues is *Functional Requirement Decision* (67.6%), followed by *Architecture Decision* (11.1%). The rest types of decisions are all below 10%. 5.4% of the decisions belong

TABLE I: A classification of decisions made in the Budibase closed issues

Type		Description	Example	Percentage
Requirement Decision	Functional Requirement Decision	A description of the decisions on services that a software system is supposed to accomplish involving calculations, technical details, data manipulation and processing, and other specific functionality.	*As a user, I want to be able to configure more advanced searching capabilities on the search component, so that I can find what I am looking for with a higher degree of accuracy.*	67.6%
	Non-functional Requirement Decision	A description of the quality attributes of a software system, judging the software system based on non-functional standards that are critical to the success of the software system.	*When I'm setting up a relationship in budibase, the current relationship configuration UI is confusing and difficult to remember how to use, I want to have a simpler and more consistent experience with internal relationships so it's easier to set up.*	5.4%
Architecture Decision		A description of the decisions with regard to architectural additions, subtractions, and modifications to the software architecture, the rationale, and the design rules, design constraints, and additional requirements that (partially) realize one or more requirements on a given architecture [20].	*Replace AppImage with Snap and RPM installations for linux.*	11.1%
Management Decision	Version Control Decision	A description of the decisions on tracking and managing changes to source code in version control systems.	*I noticed in the input component that it was still using the old css vars, these need to be updated.*	1.6%
	Documentation Decision	A description of the decisions related to documentation, e.g., changes to README files, docs, and GitHub discussions, etc.	*Stage 1 Going to add code documentation of all functions, in core, server and builder. Stage 2 (maybe a future issue) Use document code to produce markdown files, and publish to GitBook @ apidocs.budibase.com (does not exist yet).*	3.2%
Build Decision	Source Code Decision	A description of the decisions on making changes to source code.	*Create a new Options type in the backend/constants file. Remove the Categories values list from the string type in CreateEditColumn. Account for the new options field everywhere that we are checking for field.constraints.inclusion.*	2.2%
	Continuous Integration Decision	A description of the decisions concerning the compilation and building in CI pipeline supported development.	*Have a rollback mechanism on that CI job, so that we can roll back to the previous release, or a pre-specified version.*	0.3%
Testing Decision	Bug Fixing Decision	A description of the decisions made to fix the problem when a bug is generated.	*This fix for this is simply parsing the data in data providers for any datetime fields and building the lucene query with ISO strings like before.*	4.6%
	Traditional Testing Decision	A description of the decisions on examining the artifacts and the behavior of a software system under test by validation and verification.	*When testing automations on the platform, I want to be able to see the data flow from one block to the other so that I can identify the inputs/outputs of each block and debug with more ease.*	4.3%
	Continuous Integration Testing Decision	A description of the decisions with respect to the testing in CI pipeline supported development.	*Individual unit tests for each automation block. Ability to test automation integration, i.e., provide automation schema and make sure it runs successfully.*	0.3%
Deployment Decision	Traditional Deployment Decision	A description of the decisions regarding all the activities that make a software system available for use [21].	*When a user deploys a budibase application, we need to update data correctly in our dynamoDB database so that users deployment quotas are managed correctly, and that deployments will fail if they are going to go over the current quota for a particular account.*	4.3%
	Continuous Deployment Decision	A description of the decisions made when code changes are automatically deployed to a production environment through a pipeline as soon as they are ready, without human intervention [22].	*Automated CI pipeline to deploy the latest helm chart to the pre-prod environment.*	1.4%

to *Non-functional Requirement Decision* while 4.6% of the decisions are *Bug Fixing Decision*. The percentages of *Traditional Testing Decision* (4.3%) and *Traditional Deployment Decision* (4.3%) are the same. The percentages of *Continuous Integration Decision* (0.3%) and *Continuous Integration Testing Decision* (0.3%) are the least. Note that, since one issue may contain multiple types of decisions, the sum of the percentages of all types of decisions is greater than 100%.

V. Discussion

We first explain the results of this study and discuss their implications for practitioners and researchers. We then compare the study results, i.e., decisions made in CI/CD, with the decisions made in traditional open source development without using CI/CD.

A. Types of Decisions Made in CI/CD

1) Interpretations: The Software Development Life Cycle (SDLC) simply outlines the tasks required to put together a software application. In the planning phase, requirements are defined to determine what the application is supposed to do and what quality attributes need to be met. Therefore, it can be seen from the statistics that, 67.6% of the decisions are for functional requirements and 5.4% are for non-functional requirements, which is reasonable that functional requirements are the major part to make a decision in requirements analysis. Then the creation phase models the way a software application will work. Architecture design is part of the outcome of this phase, and our result shows that 11.1% of the issues contain architecture decisions. Based on the architecture design, developers start the actual implementation of the application. Usually, an open source project is implemented by a virtual team and the implementation tasks can be broken down into jobs for each developer, and consequently a source code management tool is used to help developers track changes to the code. We found that 1.6% of the issues contain version control decisions. During the coding process, developers do not just code, they also write instructions and explanations about the code and developed application. Documentation,

such as user guides, is written to give users a quick tour of the application's basic features, while comments in the source code provide further information about the code for other developers. In our study, we found that 3.2% of the decisions are related to documentation. Before making an application available to users, it is critical to test it, and 4.3% of the traditional testing decisions are identified in the result. When an error occurs, we need to decide how to fix it, and we can see 4.6% of the decisions belonging to bug fixing. In the deployment phase, the application is made available in the user or production environment. 4.3% of the issues are relevant to traditional deployment decisions. In the CI/CD pipeline, one advantage is that some tasks in SDLC, especially integration and deployment, can be automated. Our statistical result shows that 0.3% of the issues belong to continuous integration decisions, 0.3% of the issues contain continuous integration testing decisions, and 1.4% of the decisions are about continuous deployment.

As we can observe from the results, a large percentage of decisions are made during the planning and creation phase of software development, followed by bug fixing and testing phrase. Given that the subject of the study (the selected project Budibase) is a CI/CD pipeline supported development project, we only got a small number of CI/CD-related decisions (e.g., *Continuous Deployment Decision* and *Continuous Integration Decision*). The possible reasons of this finding are that developers in CI/CD still focus on the traditional decision types and CI/CD only provides support to the project (e.g., by CI/CD pipeline tools) without the necessity of making many CI/CD-related decisions, or CI/CD-related decisions are not discussed and made in GitHub Issues, but in GitHub Actions, which requires further investigation.

2) Implications: For practitioners, our result suggests that they should pay more attention to the management, build, testing, and deployment decisions as they are as equally important as requirements and architecture decision in CI/CD. Practitioners should also put more effort on the issues and making decisions related to CI/CD, such as *Continuous Integration Decision*, which are fundamental decisions to facilitate the CI/CD pipeline.

For researchers, we provide a dataset of various types of decisions made in CI/CD, which can help to further explore decision-making in CI/CD, as well as the difference compared to decision making in traditional software development [11]. It is also interesting to further explore other data sources that communicate and make decisions in CI/CD and may partially answer the question why there are not many CI/CD-related decisions in issues.

B. Differences from Traditional Software Development

We further compared our results (i.e., decision types made in CI/CD) with the study results by Li *et al.* in our previous work [11] (i.e., decision types made in traditional open source development without using CI/CD) in this section.

From the perspective of decision categories, although there are differences in descriptions between our major decision categories and the classification of decisions in the Hibernate developer mailing list, they are similar in meaning and both basically cover the entire life cycle of development. The decision types we got in this work are more complete with one more category (i.e, *Deployment Decision*). In the decision subcategories, we did not identify *Model Decision*, *Pattern Decision*, *Development Criteria Decision*, *Implementation Decision*, and *Annotation Decision*, but we got four more decision subcategories: *Continuous Integration Decision*, *Continuous Integration Testing Decision*, *Traditional Deployment Decision*, and *Continuous Deployment Decision*, which are largely related to CI/CD.

In terms of the percentages of various types of decisions, the largest proportion of decision type in traditional software development [11] is *Design Decision* (42.6%), followed by *Requirement Decision* (31.6%). The percentages of *Management Decision* (10.1%) and *Construction Decision* (9.8%) are roughly the same, and the percentage of *Test Decision* (5.9%) is the least. However, the largest share of decision type in CI/CD is *Requirement Decision* which accounts for 73.0%, followed by *Architecture Decision* (11.1%) and *Testing Decision* (9.2%). For the remaining three decision types, each accounts for less than 6.0%.

Although these two studies used different data sources (i.e., issues in this work and developer mailing lists in [11], respectively) to investigate the decisions made, *Requirement Decision* and *Design (Architecture) Decision* are dominant in both studies, and the percentages of *Management Decision* and *Construction Decision* in traditional development are higher than *Management Decision* and *Build Decision* in CI/CD, which is reasonable since the CI/CD pipeline handles most of management and build (construction) issues. *Deployment Decision* only exists in CI/CD which is implied in the name of continuous deployment.

VI. Threats to Validity

We discuss the potential threats to the validity of our results below by following the guideline in [23]. Internal validity is not discussed because this aspect of validity is of concern when casual relationships are examined [23], and we did not investigate any causal relationships in our study.

Construct validity concerns generating the results of the study using the concept or theory behind the study [23]. In our study, one potential threat to this validity comes from manual extraction and analysis of data. To migrate this threat, we randomly selected 50 issues from the dataset and discussed the criteria for data filtering, extraction, and analysis. Before the formal data extraction and analysis, the first and second authors had agreed on the criteria, and any uncertainty was discussed among the author to reach a consensus and eliminate personal bias. Another threat is that we did not consider whether the project is born with a CI/CD pipeline, and the statistics on the percentage of CI/CD decisions may be affected by the selection of the project that started using CI/CD late in the project life cycle.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [23]. In this study, we analyzed the closed issues coming from a popular repository on GitHub, Budibase, which employs GitHub Actions to support CI/CD. One threat stems from the selection of the project and data source (issues), since it is possible that some other projects using other CI/CD tools (e.g., Travis CI) may also have relevant decisions in other data sources (e.g., pull requests). So our selected repository and data source may not be representative to all the CI/CD projects and data sources. To partially mitigate this threat, we conducted pilot project search with a set of criteria, and we also plan to include projects supported by other CI/CD tools with diverse data sources in our next step. Another threat is caused by the fact that we targeted one project only, and we will cover more repositories to migrate this threat.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers [23]. One threat could be that the first author completed all the data extraction and analysis, and discussed the uncertain data with the second author to reach an agreement, which may lead to bias during data extraction and analysis. To alleviate this threat, we detailed the research process in Section III, and the dataset and analysis results from the study have been made available online [19]. With the measures stated above, we are confident that the study results are relatively reliable.

VII. Conclusions and Future Work

As a set of established software quality assurance and development practices, continuous integration and continuous delivery are of interest to many researchers and practitioners, while the decisions made in CI/CD are not well understood. We conducted an exploratory study to obtain the types of decisions made in CI/CD using the closed issues (1,168) of Budibase as our dataset. After data filtering and extraction, we got 370 issues that contain decisions out of the 1,168 issues for further analysis. The findings of this study are the following: (1) We got 6 main categories and 11 subcategories of decisions made in CI/CD, in which *Requirement Decision* has the highest percentage (67.6%), with only a small amount of decisions (e.g., *Continuous Deployment Decision*) related to CI/CD. (2) The types of decisions made in CI/CD we obtained in this study have certain common points with the types of decisions made in traditional software development, while *Deployment Decision* only exists in CI/CD. (3) Practitioners are encouraged to put more effort on the issues and making decisions related to CI/CD, and researchers can extend the dataset and decision types got from this study by exploring projects employing other CI/CD tools and diverse data resources.

In the next step, we plan to extend this work on studying decisions made in CI/CD with a larger dataset from more repositories and diverse sources, and using complementary research methods (e.g., questionnaire, interview, and focus group). We also intend to take a deeper look at various aspects of decisions in CI/CD, including: (1) software artifacts involved in CI/CD-related decisions; (2) the rationale behind CI/CD decision-making; and (3) the impact of CI/CD-related decisions in development.

References

[1] K. Gallaba and S. McIntosh, "Use and misuse of continuous integration features: An empirical study of projects that (mis)use Travis CI," *IEEE Transactions on Software Engineering*, vol. 46, no. 1, pp. 33–50, 2020.

[2] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "The impact of continuous integration on other software development practices: A large-scale empirical study," in *Proceedings of the 32nd ASE*. IEEE, 2017, pp. 60—-71.

[3] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *Proceedings of the 31st ASE*. ACM, 2016, pp. 426—437.

[4] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[5] M. Shahin, M. Zahedi, M. A. Babar, and L. Zhu, "An empirical study of architecting for continuous delivery and deployment," *Empirical Software Engineering*, vol. 24, pp. 1061—1108, 2019.

[6] D. G. Widder, M. Hilton, C. Kästner, and B. Vasilescu, "A conceptual replication of continuous integration pain points in the context of Travis CI," in *Proceedings of the 27th ESEC/FSE*. ACM, 2019, pp. 647—-658.

[7] A. Shahbazian, Y. Kyu Lee, D. Le, Y. Brun, and N. Medvidovic, "Recovering architectural design decisions," in *Proceedings of the 15th ICSA*. IEEE, 2018, pp. 95–104.

[8] T. Olsson, K. Wnuk, and T. Gorschek, "An empirical study on decision making for quality requirements," *Journal of Systems and Software*, vol. 149, pp. 217–233, 2019.

[9] M. Shahin, P. Liang, and M. R. Khayyambashi, "Architectural design decision: Existing models and tools," in *Proceedings of the Joint WICSA/ECSA*. IEEE, 2009, pp. 293–296.

[10] M. L. Drury-Grogan, "Performance on agile teams: Relating iteration objectives and critical decisions to project management success factors," *Information and Software Technology*, vol. 56, no. 5, pp. 506–515, 2014.

[11] X. Li, P. Liang, and T. Liu, "Decisions and their making in OSS development: An exploratory study using the hibernate developer mailing list," in *Proceedings of the 26th APSEC*. IEEE, 2019, pp. 323–330.

[12] A. Shahbazian, Y. K. Lee, Y. Brun, and N. Medvidovic, "Making well-informed software design decisions," in *Proceedings of the 40th ICSE Companion*. ACM, 2018, pp. 262—-263.

[13] PMI, "Capturing the Value of Project Management through Decision Making," 2015.

[14] A. Tang and R. Kazman, "Decision-making principles for better software design decisions," *IEEE Software*, vol. 38, no. 6, pp. 98–102, 2021.

[15] P. N. Sharma, B. T. R. Savarimuthu, and N. Stanger, "Extracting rationale for open source software development decisions: A study of python email archives," in *Proceedings of the 43rd ICSE*. IEEE, 2021, pp. 1008–1019.

[16] T. Liu, P. Liang, C. Yang, Z. Xiong, C. Wang, and R. Li, "Understanding the decision-making of students in requirements engineering course projects." in *Proceedings of the 2nd SEED*. CEUR-WS, 2019, pp. 1–8.

[17] V. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," *Encyclopedia of Software Engineering*, pp. 528–532, 1994.

[18] B. G. Glaser, "The constant comparative method of qualitative analysis," *Social Problems*, vol. 12, no. 4, pp. 436–445, 1965.

[19] Y. Luo, P. Liang, M. Shahin, Z. Li, and C. Yang, "Dataset of the Paper "Decisions in Continuous Integration and Delivery: An Exploratory Study"," 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6360830

[20] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *Proceedings of the 5th WICSA*. IEEE, 2005, pp. 109–120.

[21] S. P. Roger and R. M. Bruce, *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 2015.

[22] C. O. Matthew Skelton, *Continuous Delivery with Windows and .NET*. O'Reilly Media, Inc., 2016.

[23] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

A THG Performance Case Study in the world of E-Commerce

Philip Wilson, Rehman Arshad, James Creedy, Adam Dad, Eloise Slater, Hannah Cusworth

philip.wilson, rehman.arshad, adam.dad, eloise.slater, hannah.cusworth @thehutgroup.com

The Hut Group
Chicago Ave, Voyager House, Manchester, UK

Abstract

THG's in-house e-commerce platform includes a micro-service aggregator and server-side template renderer (that we call THG Aggregator for this paper) that handles hundreds of THG websites. It is not uncommon in e-commerce to entertain thousands of requests per minute and THG Aggregator uses extensive in-memory (on JVM) caching to support the performance requirements. This case study revolves around analysing the performance of this in-house system on current JDK version (JDK 8) it is running on and the latest LTS (long term support) version (JDK 11) to decide the LTS version configurations before migrating THG Aggregator to that. We analysed parameters like heap utilisation, GC (garbage collector) pause time and process usage under different configurations of JDK 8 and 11. Based on this analysis, we showed different options and configurations of JVM that can enhance or decrease the performance of THG Aggregator. We have also conducted extensive analysis of all these variations on Arm VS Intel to extract the best combination of instruction set architecture and JDK variation in terms of the response time of user requests.

Key Words —E-Commerce, JDK, JVM [1]

I. Introduction

THG Aggregator is the heart of THG's e-commerce operation (Fig. 1). THG Aggregator handles hundreds of THG websites and act as a major caching and aggregating entity between the outside world and the internal resources. This case study revolves around analysing the performance of THG Aggregator on current JDK version (JDK 8) it is running on and the latest LTS (long term support) version (JDK 11) to decide the LTS version configurations before migrating THG Aggregator to that. We have defined a framework of analysis that can show variations in the values of parameters like heap usage, total duration of GC (garbage collector) pauses, app usage etc. based on different configurations of JVM. Configuration parameters include parallel GC threads (*-XX:ParallelGCThreads*), type of garbage collector, maximum heap size and *Stringdeduplication* etc.

In order to make sure that the patterns of the findings hold for different number of requests, we have conducted analysis based on 1, 10, and 100 and *n* number of requests whereas, *n* is the number of requests set via load testing. Table I shows different configurations used in the analysis, ranges from v0-v8. The configuration v4 is currently being used in THG with Java 8. We have compared v0-v4 via our framework of analysis between Java 8 and 11 to show the performance enhancements by hitting the running configurations of THG Aggregator. From v5-v8, the configurations were only run against Java 11 to analyse if these configurations can provide better performance as compared to v4. Few parameters can also show some variation based on the current usage and specifications of a machine therefore, we have conducted each type of request on multiple machines and use the average values in our framework of analysis. All the virtual and physical machines used in this analysis had the same specifications to rule out any inconsistency in data.

The instances of THG Aggregator used in this research are not set-up with load balancers and are based on a single proxy server therefore, actual values of defined parameters for system in production are way more optimised however, the test bed created for this analysis is enough to compare and contrast the different configurations and load testing for the required analysis.

The remainder of this paper is organised as follows:

[1]DOI reference number: 10.18293/SEKE2022-067

GC Configurations	
v0	Default Configurations of a JDK Version
v1	-Xmx10240M -Xms10240M
v2	-Xmx10240M -Xms10240M -XX:+UseG1GC
v3	-Xmx10240M -Xms10240M -XX:+UseG1GC -XX:+UseStringDeduplication
v4	-Xmx10240M -Xms10240M -XX:+UseG1GC -XX:+UseStringDeduplication -XX:InitiatingHeapOccupancyPercent=60
v5	-Xmx10240M -Xms10240M -XX:+UseG1GC -XX:+UseStringDeduplication -XX:InitiatingHeapOccupancyPercent=70
v6	v4 + -XX:ParallelGCThreads=16 -XX:ConcGCThreads=4
v7	v4 + -XX:G1MixedGCLiveThresholdPercent=75
v8	v5 + -XX:ParallelGCThreads=16 -XX:ConcGCThreads=4 -XX:G1MixedGCLiveThresholdPercent=75

TABLE I: GC Configurations for THG Aggregator

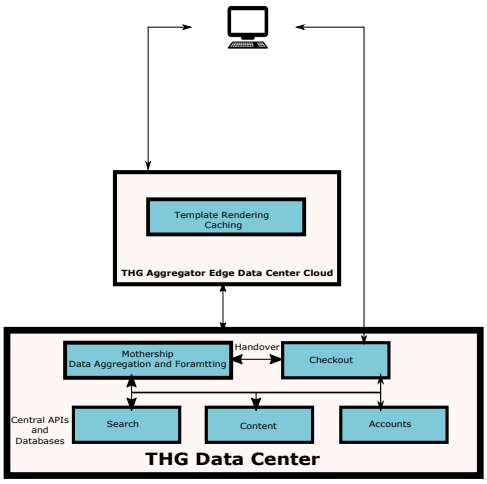


Fig. 1: THG Aggregator

Section II defines the framework of analysis used for comparing the performance of THG Aggregator. Section III discusses the findings and the analysis. This section shows multiple tables and graphs to compare and contrast the running configurations on Java 8 and 11. Section IV includes the related work which is confined to similar studies and case reports. Section V is the last section and it includes conclusion and future work.

This paper can be used as a reference in the world of e-commerce to conduct performance based analysis of different versions of JVMs under different configurations. Some pre-planned analysis like this can ensure the benefits of migration under different set of configurations to latest LTS version of Java.

II. Framework of Analysis

Based on the configurations defined in Table. I, we have divided our analysis framework into two main parts.

- First part is about analysing the important JVM parameters. Table. II is showing parameters of part 1 of analysis framework, and they are defined as follows:
  - ***GC Young Config.*** refers to the garbage collector used for young generation e.g., in JDK 8, default one is parallel scavenger whereas, in JDK 11, default garbage collector is G1 New.
  - ***GC Old Config.*** refers to the garbage collector used for old space.
  - ***GC Time Ratio.*** refers to the ratio between the time spent in GC and the time spent outside of GC.
  - ***Max. Allocated Heap.*** refers to the amount of heap set as the maximum value (-Xmx).
  - ***Max. Young generation size.*** refers to the size allocated to young generation.
  - ***Max. heap used.*** refers to the maximum heap consumed by a specific configuration on specific JDK version.
  - ***Max. heap value post GC.*** refers to the heap memory a specific configuration holds after a major GC.
  - ***Total duration of GC pauses.*** refers to the sum of all the GC pauses that take place during n number of requests under specific configurations and JDK version.
  - ***Longest Pause.*** refers to the longest pause value out of all the pauses GC takes under specific configurations and JDK version.
  - ***CPU Usage: Machine.*** refers to total utilisation of the machine's CPU while running an instance of THG Aggregator under specific configurations and JDK version.
  - ***CPU Usage: JVM + App.*** refers to the percentage of CPU utilisation takes by THG Aggregator and JVM out of the total percentage of machine usage e.g., if machine usage is 89.2% and *JVM+App* usage is 24.38% then THG Aggregator is taking 27.33% of the usage under that configuration.
- Second part (Part 2) is related to load testing i.e., the number of requests a specific configuration of THG Aggregator can handle under defined time. We ran *n* number of requests directed to running instances of THG Aggregator for *m* number of users and compare

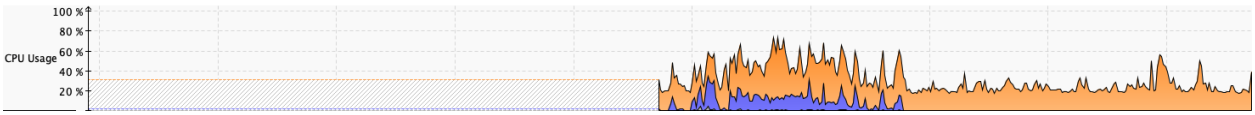


Fig. 2: Machine and *JVM+App* Usage: THG Aggregator 100 requests using JDK 11

the results handled by different configurations of JVM to analyse the impact of these configurations on performance of THG Aggregator. The performance will be elaborated via tables and graphs in section III-B.

100 Requests (no-cache) v1 Avg.		
	JDK 8	**JDK 11**
GC Young Config.	Parallel Scavenger	G1 New
GC Old Config.	Parallel Old	G1 Old
GC Time Ratio	99%	12%
Max. allocated heap	10 GB	10 GB
Max. Young generation size	3.33 GB	1.3mb
Max. heap used	3.18 GB	1.56 GB
Max. Heap value post GC	478mb	451.5mb
Total duration of GC Pauses	2828.18ms	885.73ms
Longest Pause	1298ms	281.49ms
Max CPU Usage: Machine	89.2%	84.4%
Max CPU Usage: JVM+App	24.38%	24.3%

TABLE II: THG Aggregator on JDK 8 VS JDK 11: Avg. of 100 Requests based on 2 different machines using v1

100 Requests (no-cache) v2 Avg.		
	JDK 8	**JDK 11**
GC Young Config.	G1 New	G1 New
GC Old Config.	G1 Old	G1 Old
GC Time Ratio	9%	12%
Max. allocated heap	10 GB	10 GB
Max. Young generation size	1.3mb	1.3mb
Max. heap used	2.08 GB	1.71 GB
Max. Heap value post GC	572mb	557.5mb
Total duration of GC Pauses	1210.19ms	1076ms
Longest Pause	251.11ms	332.59ms
Max CPU Usage: Machine	90.2%	73.7%
Max CPU Usage: JVM+App	38.5%	36.45%

TABLE III: THG Aggregator on JDK 8 VS JDK 11: Avg. of 100 Requests based on 2 different machines using v2

Overall, first part of the analysis was conducted with variable number of requests directed to running instances of THG Aggregator with defined configurations of specific JDK version. The final values of the defined parameters are the averages taken from running the same configurations on multiple machines under same load. Same set of requests was used for all the combinations in order to be consistent across different JDKs and configurations. The second part of the analysis was conducted against various sets of n requests with m users spawn up every second to see how JVM based optimisations can impact the end user experience by offering better performance.

100 Requests (no-cache) v3 Avg.		
	JDK 8	**JDK 11**
GC Young Config.	G1 New	G1 New
GC Old Config.	G1 Old	G1 Old
GC Time Ratio	9%	12%
Max. allocated heap	10 GB	10 GB
Max. Young generation size	1.3mb	1.3mb
Max. heap used	1.91 GB	1.97 GB
Max. Heap value post GC	602.5mb	513.5mb
Total duration of GC Pauses	1501ms	759ms
Longest Pause	380.28ms	212.37ms
Max CPU Usage: Machine	89.3%	64.7%
Max CPU Usage: JVM+App	14.9%	18.5%

TABLE IV: THG Aggregator on JDK 8 VS JDK 11: Avg. of 100 Requests based on 2 different machines using v3

100 Requests (no-cache) v4 Avg.		
	JDK 8	**JDK 11**
GC Young Config.	G1 New	G1 New
GC Old Config.	G1 Old	G1 Old
GC Time Ratio	9%	12%
Max. allocated heap	10 GB	10 GB
Max. Young generation size	1.3mbGB	1.3mb
Max. heap used	3.31 GB	1.62 GB
Max. Heap value post GC	611.5mb	522mb
Total duration of GC Pauses	1243.68ms	706.70ms
Longest Pause	459.2ms	190.5ms
Max CPU Usage: Machine	76.5%	69.75%
Max CPU Usage: JVM+App	40.7%	25.45%

TABLE V: THG Aggregator on JDK 8 VS JDK 11: Avg. of 100 Requests based on 2 different machines using v4

III. Findings and Discussion

A. Analysis of JVM Parameters

For part 1 of the analysis and configuration v1, Table. II is showing 100 requests directed to running instances of THG Aggregator on JDK 8 and 11 respectively. In the stated table, default configurations of these JDK versions were used i..e., JDK 8 runs on *parallel scavenger* and *parallel old* GC configurations VS G1 New and G1 Old GC configurations of JDK 11. For both JDKs, maximum allocated heap was 10 GB. JDK 11 used way less heap than JDK 8 which is due to better heap management in JDK 11. Overall, JDK 11 performed considerably better

THG Aggregator 100 Requests (no-cache) JDK 11 Based on Avg. of Two Machines					
	V4	**V5**	**V6**	**V7**	**V8**
GC Young Config.	G1 New	G1 New	G1 New	G1 New	G1 New
GC Old Config.	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old
GC Time Ratio	12%	12%	12%	12%	12%
Max. allocated heap	10 GB	10 GB	10 GB	10 GB	10 GB
Max. Young generation size	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb
Max. heap used	1.62 GB	1.32 GB	1.54 GB	1.48 GB	3.41 GB
Max. Heap value post GC	522mb	514.5mb	356.5mb	552mb	530.5mb
Total duration of GC Pauses	706.70ms	1271.56ms	512.38ms	1374ms	2690.27ms
Longest Pause	190.5ms	356.24ms	242ms	398.5ms	1512.2ms
Max CPU Usage: Machine	69.75%	71.1%	68.2%	90%	78%
Max CPU Usage: JVM+App	25.45%	48.15%	51.45%	50.95%	38.35%

TABLE VI: THG Aggregator on JDK 11: 100 Requests based on different GC Configurations

in heap management, pause times and usage VS JDK 8.

For the configuration v2 (both JDK 8 and 11 were using G1 in v2) Table. III showed that JDK 11 was still better in GC total pause duration however, *JVM+App* utilisation was 42.68% (i.e., 38.5 is 42.68% of 90.2 which is total CPU usage in this case) of the machine usage in JDK 8 VS 49.45% in JDK 11 (for 100 requests) i.e., much improvement was seen in JDK 8 with G1 though overall, JDK 11 still performed better. All this data was recorded and profiled via *flight recorder* and *async profiler*. Table. IV showed that the gap between JDK 8 and 11 further narrowed down in terms of the stated parameters e.g., for 100 requests, *JVM+App* utilisation was 16.68% in JDK 8 VS 28.59% in JDK 11 i.e., 8 performed better than 11. Heap utilisation is also neck-to-neck (heap utilisation is slightly better with JDK 8 100 requests with v3). The only major difference left between JDK 11 and 8 is the total duration of GC pauses (759ms VS 1501ms in 100 requests) that is still far apart in 8 and 11.

For the configuration v4 (Table. I), it was observed that *JVM+App* usage was better in JDK 8 (difference of 14%) with 10 requests but as Table. V showed, JDK 11 took the considerable lead of more than 16% in 100 requests. Better heap utilisation and GC pause times validated that running THG Aggregator on the configuration v4 with JDK 11 is better than the current running configurations (v4 with JDK 8). From v1-v4, trend is pretty consistent in favour of JDK 11 and overall, JDK 11 always performed better than JDK 8.

It is pretty evident from the analysis so far that JDK 11 is pretty consistent in performing better than JDK 8 except few occurrences where JDK 8 performed slightly better in CPU utilisation therefore, we defined configurations v5-v8 to find out if these configurations can perform better than the configuration v4 on JDK 11. Table. VI is showing our analysis for configurations v4-v8 on JDK 11 for 100 user requests directed to running instance of THG Aggregator with stated configurations. These configurations were selected by combining the JVM flags that can help in reducing pause time, better heap utilisation and better machine usage.

According to Table. VI, configuration v6 showed best GC pause duration value of 512.38ms VS 706.70ms of v4. V6 also showed minimum value for the longest pause. The heap utilisation of v6 is slightly higher than v5 and v7. v6's *JVM+App* utilisation is 75.43% of the machine utilisation VS 36.48% in v4 which is the minimum value in the stated configurations. V6 is set up with 16 *ParallelGcThreads* and 4 *ConcGCThreads* as compared to 8 *ParallelGcThreads* and 2 *ConcGCThreads* in v4 therefore, v6 is utilising more CPU and a little bit more heap but producing the best results in terms of GC pause duration that can be really useful in case of thousands of requests at the price of higher machine utilisation that demands better hardware overall. V8 was the worst performer (v8 is the aggregation of all the GC flags from other configurations) with highest GC pause time, 49.16% of machine utilisation and 3.41 GB of heap utilisation. These results clearly stated that from GC perspective, v6 is the best configuration at the price of higher CPU utilisation and v4 has better machine utilisation at the expense of higher values for GC pause duration and heap utilisation. Overall, JDK 11 always performed better with v4 as compared to JDK 8 in terms of heap utilisation and GC total pause duration.

We went one step further and also analysed and compared 100 requests on running instances of THG Aggregator on *Arm* and *Intel*. For that purpose, we used *amazon linux2* on both instances with centos 7. One instance was

THG Aggregator 100 Requests (no-cache) JDK 11 AWS Graviton VS Intel																		
	V0		V1		V2		V3		V4		V5		V6		V7		V8	
	Arm	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*	*Arm*	*Intel*
GC Young Config.	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New	G1 New
GC Old Config.	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old	G1 Old
GC Time Ratio	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
Max. allocated heap	3.89 GB	3.89 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB
Max. Young generation size	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb	1.3mb
Max. heap used	998mb	877mb	1.73GB	1.43GB	1.46 GB	3.11GB	2.56 GB	1.45GB	1.54 GB	1.49GB	2.07 GB	1.51GB	2.55 GB	2.05 GB	3.79 GB	1.26 GB	2.35 GB	1.57 GB
Max. Heap value post GC	614mb	622mb	562mb	594mb	606mb	568mb	560mb	626mb	562mb	602mb	595mb	615mb	625mb	584mb	572mb	616mb	484mb	607mb
Total duration of GC Pauses	742.70ms	1187.77ms	722.76ms	1264.38ms	800.50ms	1030ns	796.70ms	1319.17ms	865.89ms	1095.06%	901.35ms	1260.59ms	922.78ms	1356.65ms	686.55ms	1294.8ms	989.95ms	1292.47ms
Longest Pause	80.63ms	95.49ms	153.55ms	201.02ms	179.43ms	254.93ms	242.12ms	259.72ms	240.76ms	222.30ms	208.28ms	304.54ms	283.64ms	260.95ms	177.28ms	195.47ms	265.26ms	259.50ms
Max CPU Usage: JVM + App	84.3%	92.7%	84.7%	94%	86.6%	93.8%	86.1%	91.3%	86.2%	93%	89.1%	95.8%	85.4%	94%	86.6%	99.3%	84%	96.3%

TABLE VII: THG Aggregator 100 Requests: JDK 11 (V0-V8) AWS Graviton VS Intel

running on Arm Graviton and the other one was on intel X86 architecture. Both instances spinned up with 4 vcpus and 16GB RAM. No other service was running on these instances during the profiling i.e., THG Aggregator was the only source of machine usage other than some basic OS services. Table. VII is showing the summary of this analysis. Overall, in all the configurations from v0-v8, machine utilisation was always higher in Intel as compared to Arm. Intel showed better readings for heap utilisation e.g., in v4, v6, v7 and v8 but this parameter alone cannot compensate the huge difference in total duration of GC pauses. In case of v6, the difference is 922.78ms(Arm) VS 1356.65ms(Intel) therefore, Arm in general and Arm v6 in particular still came out as the best overall configuration for THG Aggregator.

The parameters stated above are important in identifying the performance of a configuration but in the world of e-commerce, one important parameter of performance is the ability to entertain specific number of requests under defined time frame therefore, next section will evaluate THG Aggregator on v6 and v4 (on Arm based machines) to analyse how these parameters will be translated into actual performance of an e-commerce system.

B. Performance Analysis Via Load Test

THG Aggregator Load Test (Arm)						
	v4	**v5**	**v6**	**v7**	**v8**	**v6 Intel**
Time	10 mins	10 mins	10 mins	10 mins	10 mins	10 mins
Total Reqs.	52589	52235	53458	53008	52219	28335
Max. RPS	162	165.6	184.4	170.5	197.9	104.2
Avg. RPS	87.6	86.9	88.9	88.2	86.9	47.2

TABLE VIII: THG Aggregator Load Test: Max. 800 Users

For the load testing, the requests were ranging from simple user *GET* end points to the requests related to checkout basket etc. For all configurations, load test was conducted for 10 minutes, starting from zero users, spawning 10 users each second with maximum user limit to 800. We used THGs *Graviton*(in-house load tester) and *Locust* [10] for running the load test and getting the required data. Table. VIII is showing the results of our load test.

In the stated table, **Time** shows total duration for which we ran the load test, **Total Reqs.** shows number of requests entertained in defined time by a specific configuration, **Max. RPS** is showing maximum requests per second during the defined time and **Avg. RPS** is showing average requests per second during the load test. We ran the load test for configurations v4-v8 and as the table shows, v6 is a better performer i.e., better JVM metrics of v6 also translated into actual performance. V6 entertained 53458 requests with 184.4 **Max. RPS** and 88.9 **Avg. RPS**. Second closest candidate was v7 in terms of total requests but its average and maximum requests per second were lower than the v6. V8 is showing better value for **Max. RPS** but it cannot compete with v6 on other parameters and v8 was also one of the worst performers in JVM analysis.

The results were a little different with increasing number of users though e.g., when load testing was conducted for 1200 maximum users instead of 800, v4 showed slightly better results than v6 (52094 total requests on v4 VS 51870 on v6, 86.5 **Avg. RPS** on v4 VS 86.2 on v6). With 1500 maximum users, same trend between v4 and v6 still persisted as v6 has higher number of parallel and concurrent threads therefore, the slight decline in the values indicate the bottleneck in CPU usage in the running instance of THG Aggregator.

For comparison, we also ran *Intel v6* to compare it with *Arm v6* and the difference in the outcome is huge. *Intel v6* only entertained 28335 requests against 53458 in *Arm v6*. **Max. RPS** and **Avg. RPS** of *Intel v6* are also quite poor as compared to the *Arm v6* therefore, *Arm v6* seems to be the best configuration in our load test. Same trend was seen between *Arm* and *Intel* after changing user range multiple times between 400-1500. The user-range beyond 1500 was not realistic as in e-commerce, load balancers and orchestration tools are used to redirect load to different instances therefore, ceiling of 1500 was realistic for one running instance of an e-commerce platform.

IV. Related Work

There are various studies on JVM that analyse and propose options for fine tuning and optimisation. Our point of interest revolves around those approaches that investigate the performance tuning at JVM level in the domain of web-services in general and e-commerce in particular e.g., [8] investigates performance overhead of JVM in data parallel systems, [3] conducts a study to investigate ageing of JVM from memory depletion point of view and [4] looks into JVM enhancements for server-specific performance.

Few approaches are not the case studies but language oriented e.g., [5] discusses JVM from JIT (Just-In-Time) point of view to analyse JIT compiler abstraction management.

In the domain of e-commerce and micro-services, there are many case studies that include the performance enhancements and performance engineering of Java systems at JVM or framework level and these are the approaches that are closest to our approach e.g., [9] discusses the performance engineering of e-commerce systems but this approach proposes enhancement at the level of framework (EJB [2]), not at the level of JVM. [6] looks into performance enhancement of garbage collector in java based web-services. [1] proposes a simulation model to test and diagnose issues in production systems and [11] discusses the enhancements to reduce the run time overheads of JIT compiler to address the busy traffic at *Alibaba*.

Other than the optimisations based on JIT and overheads, few approaches try to tackle scalability issues via distributed JVMs. One such approach is JESSICA2 DJVM [7] that tries to cluster and scale the web application servers with distributed JVMs. *CHAOSMACHINE* [12] is another approach that does not tweak JVM itself but injects perturbations to JVM via try-catch blocks in order to extract an analysis of exception handling capabilities of a code base.

Our approach does not discuss any framework like EJB. The novelty lies in the fact that our approach involves three dimensions i.e., JVM, e-commerce and micro-services whereas, the approaches stated above involve one or two dimensions out of these three. Our approach also went one step further and analysed the implications of JVM tweaks in terms of e-commerce traffic i.e., number of requests handle by THG Aggregator under defined time frame with different configurations.

V. Conclusion and Future Work

In this paper, we presented the detailed analysis of an e-commrece system via our defined framework of analysis. The results identified the best configuration and concluded that *Arm* is the overall better performer under defined configurations to run THG Aggregator. As a part of the analysis framework, a comprehensive series of load tests showed how JVM configurations translated into actual performance of an e-commerce system.

After running and analysing THG Aggregator under different configurations, future work involves the analysis at code level to find out low-performant parts of the code base. In other words, code-based profiling should be conducted to extract call-graphs and heap dumps. Such a fine-tuning at code-level along with the JVM-based tuning will be really valuable in making an e-commerce system more robust with enhanced performance. Another dimension of future work revolves around the cost estimation of running the e-commerce operations. New *Arm* processors claim to be more energy efficient and as our results stated, performed better with THG aggregator. Further research can quantify the difference in running cost against the cost of replacing the existing hardware and further decisions can be made regarding future purchases and code optimisations against a specific hardware.

References

[1] Alberto Avritzer and Elaine J Weyuker. The role of modeling in the performance testing of e-commerce applications. *IEEE Transactions on Software Engineering*, 30(12):1072–1083, 2004.

[2] Bill Burke and Richard Monson-Haefel. *Enterprise JavaBeans 3.0*. " O'Reilly Media, Inc.", 2006.

[3] Domenico Cotroneo, Salvatore Orlando, Roberto Pietrantuono, and Stefano Russo. A measurement-based ageing analysis of the jvm. *Software Testing, Verification and Reliability*, 23(3):199–239, 2013.

[4] Robert Dimpsey, Rajiv Arora, and Kean Kuiper. Java server performance: A case study of building efficient, scalable jvms. *IBM Systems Journal*, 39(1):151–174, 2000.

[5] Malin Källén and Tobias Wrigstad. Performance of an oo compute kernel on the jvm: revisiting java as a language for scientific computing applications. In *Proceedings of the 16th ACM SIGPLAN International Conference on Managed Programming Languages and Runtimes*, pages 144–156, 2019.

[6] Hai-Shuan Lam, GSVRK Rao, Chikkanan Eswaran, and Kok-Seong Ng. Performance comparison of various garbage collectors on jvm for web services. In *2006 International Symposium on Communications and Information Technologies*, pages 711–715. IEEE, 2006.

[7] King Tin Lam, Yang Luo, and Cho-Li Wang. A performance study of clustering web application servers with distributed jvm. In *2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 328–335. IEEE, 2008.

[8] David Lion, Adrian Chiu, Hailong Sun, Xin Zhuang, Nikola Grcevski, and Ding Yuan. Don't get caught in the cold, warm-up your {JVM}: Understand and eliminate {JVM} warm-up overhead in data-parallel systems. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 383–400, 2016.

[9] T-K Liu, Santhosh Kumaran, and J-Y Chung. Performance engineering of a java-based e-commerce system. In *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004*, pages 33–37. IEEE, 2004.

[10] S Pradeep and Yogesh Kumar Sharma. A pragmatic evaluation of stress and performance testing technologies for web based applications. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 399–403. IEEE, 2019.

[11] Fangxi Yin, Denghui Dong, Sanhong Li, Jianmei Guo, and Kingsum Chow. Java performance troubleshooting and optimization at alibaba. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 11–12. IEEE, 2018.

[12] Long Zhang, Brice Morin, Philipp Haller, Benoit Baudry, and Martin Monperrus. A chaos engineering system for live analysis and falsification of exception-handling in the jvm. *IEEE Transactions on Software Engineering*, 2019.

EARS2TF: A Tool for Automated Planning Test from Semi-formalized Requirements

Hui Liu†, Yunfang Li†, Zhi Li†*
†School of Computer Science and Engineering, Guangxi Normal University, Guilin, Guangxi, P. R. China
Email: huiliu729@gmail.com, zhili@gxnu.edu.cn

***Abstract*—Software testing is critical to the integrity of the software implementation. The test engineers must design tests under the premise of ensuring the conformance of software system with respect to stakeholder requirements. Generally, requirements documents are large and requirements specifications are represented by NL(natural language) which can be error-prone, so test plans tend to consume more time and effort. In this paper, we propose a tool, EARS2TF, which supports test framework generation from NL requirements specifications. Requirements specifications are represented by a requirements template EARS (Easy Approach to Requirements Syntax), which is easy to use and capable of representing stakeholder requirements with less ambiguity. To save testing costs, we provide a technique and tool that allows requirements to be written, edited, and checked for conformance to existing requirements and EARS syntax, while allowing test engineers to test directly without validating the requirements specification. A demo video of this tool is available at https://youtu.be/fmk4xSRh40k.**

***Index Terms*—test plan, EARS, natural language requirements, requirements engineering**

I. Introduction

Software testing is the process of review of software requirements analysis, design specifications and coding throughout the whole software development lifecycle, as well as verifying the quality of the software by measuring and evaluating the quality of the software to meet stakeholder requirements [1] [2]. The test plan serves as a blueprint for testing, describing information needed to perform software product testing such as the test strategy, objectives, etc. and can help determine the information and work needed to verify the quality of the system under test.

The creation of a test plan requires a number of tasks that are currently mostly done manually and require a lot of time and effort. Test engineers need to fully understand and extract test-related information from requirements, most of which are written in natural language (NL) [3] [4] [5]. Natural language contains the inherent characteristic of ambiguity and requirements are often cumbersome, especially in large software projects, and requirements elicitation is challenging as they become complex [6]. The complexity and lack of precision of the requirements causes test planning to become a time-consuming and error-prone process, and the lack of automation in the process also leads to additional maintenance costs.

*corresponding author: zhili@gxnu.edu.cn
DOI reference number: 10.18293/SEKE2022-179

The expression of requirements has a significant impact on the creation of the testing process. Requirements that are not formally described are usually imprecise and vague, which can hinder the testing process. Creating requirements according to requirements templates, also known as Constrained Natural Language (CNL), can reduce such problems, and EARS [5] is one of the more popular templates in terms of their use in industry and their availability to practitioners [7]. The template specializes the generic requirements syntax into five types to describe requirements, is concise and clear, and demonstrates its usability with industrial examples.

In this paper, we use EARS to describe requirements to ensure their accuracy, and then automatically generate a test framework from EARS requirements to reduce the cost of testing while ensuring consistency between requirements and tests, and to provide the following tool: EARS2TF. This paper is presented as follows: Section II describes the features of EARS2TF. Section III shows the evaluation results of three cases. Section IV discusses related work. We conclude the paper in Section V.

II. Tool Features

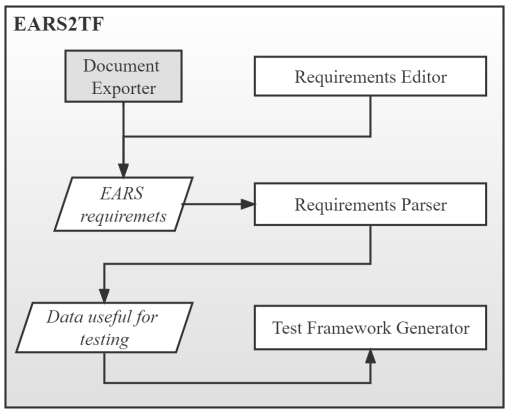


Fig. 1. EARS2TF tool architecture. Grey boxes show third-party components, while white boxes denote EARS2TF components

A. Tool Overview

EARS2TF is a testing aid tool based on Eclipse Xtext, which consists of three main parts: requirements editor, requirements parser, and test framework generator. Fig. 1 shows

the tool architecture. We design our own EARS syntax rules, requirements conforming to EARS syntax rules can be imported from existing files with the .ears suffix or obtained by using out requirements editor. EARS requirements can be parsed by the requirements parser to obtain data useful for testing, and then the test framework generator generates a test framework corresponding to the requirements based on the obtained data.

B. Requirements Editor

In general, requirements are often written with some errors that can lead to testing errors. To make requirements descriptions more accurate, EASR2TF provides a requirements editor that describes the syntactic structure of EARS using the Xtext framework. Our requirements editor enables the written requirements to conform to the EARS syntax and provides a more convenient use experience.

C. From Requirements to Test Framework

We designed the test framework by referring to some testing standards, such as ISO/IEC 29119 [8], UTP (UML Testing profile) [9], etc. A test framework is a plan for the testing process, which contains the test object, test architecture, etc. In order to narrow the gap between requirements and tests, also to avoid incomplete understanding of requirements leading to missing tests, EARS2TF can convert EARS requirements into test frameworks by designing algorithms to read key information of EARS requirements and then parse them into corresponding test information, and finally the test framework can be displayed in the form of text. Testers can then start testing directly from the test framework, avoiding spending too much time on requirements and missing test information.

III. Evaluation

We evaluated EARS2TF by using two examples from a dataset of public requirements document-PURE [10], which containing 44, 192 requirement statements, respectively. We checked and analyzed the results, and almost all of the generated test plan messages are correct except for 13 messages. The reason for these error messages is that there are 6 requirement statements that are untestable assumptions or semantically incorrect.

IV. Related Work

A lot of existing research has been devoted to automated testing, such as automatic generation of test cases, test scripts, etc from NL requirements or restricted requirements [11] [12] [13] [14]. However, none of these tasks can guarantee correct results, and some require manual addition of files or manual intervention to get the correct results. Some approaches focus on generating test model or test plan. Fischbach et al [6] generate test model from semi-structured requirements by extracting Cause-Effect-Graphs. Lukose et al [15] proposed a tool for guiding a software tester in generating test plans. All these approaches either do not automatically generate test plans or the results obtained cannot plan test. To the best, EARS2TF is the only approach that automates the test planing from requirements comply with template, and also allows requirements editing.

V. Conclusion and Future Work

In this paper, we propose a automated planning test tool from EARS requirements, which includes a requirements editor allowing requirements editing, writing and checking. Future work includes the extension of our tool to check the semantic of requirements and generate test case.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (61862009), Guangxi Natural Science Foundation (2018GXNSFAA281314, Guangxi "Bagui Scholar" Teams for Innovation and Research, the Project of the Guangxi Key Lab of Multi-source Information Mining Security (Director's grant 19-A-01-02), Innovation Project of GuangXi Graduate Education(JXXYYJSCXXM-2021-005).

References

[1] J. A. Whittaker, "What is software testing? and why is it so hard?" *IEEE software*, vol. 17, no. 1, pp. 70–79, 2000.

[2] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2016.

[3] C. Ribeiro and D. Berry, "The prevalence and severity of persistent ambiguity in software requirements specifications: Is a special effort needed to find them?" *Science of Computer Programming*, vol. 195, p. 102472, 2020.

[4] M. Luisa, F. Mariangela, and N. I. Pierluigi, "Market research for requirements analysis using linguistic tools," *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, 2004.

[5] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," in *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 2009, pp. 317–322.

[6] J. Fischbach, M. Junker, A. Vogelsang, and D. Freudenstein, "Automated generation of test models from semi-structured requirements," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 263–269.

[7] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[8] ISO, "Ieee/iso/iec 29119-1-2021," 2021.

[9] OMG, "Uml testing profile, version 2.1," 2019.

[10] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 502–505.

[11] C. Wang, F. Pastore, A. Goknil, and L. C. Briand, "Automatic generation of acceptance test cases from use case specifications: an nlp-based approach," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 585–616, 2022.

[12] T. Yue, S. Ali, and M. Zhang, "RTCM: A natural language based, automated, and practical test case generation framework," *2015 International Symposium on Software Testing and Analysis, ISSTA 2015 - Proceedings*, pp. 397–408, 2015.

[13] J. Fischbach, J. Frattini, A. Vogelsang, D. Mendez, M. Unterkalmsteiner, A. Wehrle, P. R. Henao, P. Yousefi, T. Juricic, J. Radduenz *et al.*, "Automatic creation of acceptance tests by extracting conditionals from requirements: Nlp approach and case study," *arXiv preprint arXiv:2202.00932*, 2022.

[14] D. Flemström, T. Gustafsson, and A. Kobetski, "Saga toolbox: Interactive testing of guarded assertions," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2017, pp. 516–523.

[15] K. Lukose, S. Agarwal, V. N. Rao, and J. Sreevalsan-Nair, "Design study for creating pathfinder: A visualization tool for generating software test plans using model based testing." in *VISIGRAPP (3: IVAPP)*, 2018, pp. 289–300.

A Simplified Method for Automatic Verification of Java Programs

1st Zhi Li
School of Computer Science and Engineering
Guangxi Normal University
Guilin, China
zhili@gxnu.edu.cn

2nd Ling Xie
School of Computer Science and Engineering
Guangxi Normal University
Guilin, China
lingxiehy@outlook.com

3rd Yilong Yang*
School of Software
Beihang University
Beijing, China
yilongyang@buaa.edu.cn

***Abstract*—Current KeY verification tool for Java programs provides limited capability for verifying Java programs. In order to solve this problem, we provide a method for simplifying complex Java programs into a format that is compatible with the KeY. A set of simplification rules based on abstract syntax tree (AST) are proposed. These rules can keep the logic and semantics of the original Java programs mostly unchanged, while meeting the requirements of KeY verification tool. The paper concludes with a bank ATM example to demonstrate the feasibility of our work.**

***Index Terms*—Program verification, KeY verification tool, abstract syntax tree (AST), Java program simplification**

I. Introduction

Program verification is an important part of software development, which can detect some errors in the program. Current program verification methods or techniques are only applicable to program fragments or simple programs. Therefore, a program simplification method is urgently needed to extend the capability of existing program verifiers so that complex programs can be dealt with constructively. In this paper, we provide a simplification method to enable the KeY tool[4]to verify those Java programs with moderate complexities.

The Java simplification work in this paper uses the abstract syntax tree (AST) to parse, traverse, and re-factor code. Inspired by the work of [1,2,3], AST is used to parse the variables, types and functions of the source code and then traverse and re-factor the object code, thus simplifying the Java program while keeping most of the logic and semantics unchanged.

The ultimate goal of simplifying complex Java programs is to be verifiable in the KeY tool. The KeY is a formal program tool for Java programs, with both fully automated and interactive verification. It converts the Java program as input to Java dynamic logic (JavaDL), and then verifies the JavaDL step by step applying the corresponding taclet. Finally, the verification results are presented in the form of a proof tree[4]. As a continuously improved tool, KeY supports and verifies invariant specifications. The specification and verification of invariant allows us to conveniently specify and verify strong data integrity properties for Solidity smart contracts[5].

*corresponding author: yilongyang@buaa.edu.cn
DOI reference number:10.18293/SEKE2022-180

II. Method and Implementation

A. Overview of the methods

To enable the KeY tool to verify complex Java programs, this paper presents 7 simplified rules.Details are as follows:

1) New AST Rule

$$New\ AST\ Rule\ \frac{Content}{CompilationUnit}$$

Take the source program (Content) as a parameter of the *createAST* method to generate an AST CompilationUnit.

2) Get Type Rule

$$Get\ Type\ Rule\ \frac{CompilationUnit}{TypeDeclaration}$$

With the AST as the head node, the children node is called step by step. The type of this rule includes the class name and the content of all methods, and is the header of all modification nodes.

3) Modify Data Type Rule

$$Modify\ Data\ Type\ Rule\ \frac{float}{int}$$

Modify the data type after the node location is found, then the method is called to replace the previous data type with the new data type.

4) Delete Rule

$$Delete\ Rule\ \frac{Object}{\phi}$$

The Delete rule includes deleting comments, deleting variables, deleting statements, etc. The node where the deleted object is located in the AST is found, and the *remove* or *delete* method is called to delete it.

5) Substitution Rule

$$Substitution\ Rule\ \frac{getPasswordValidated()}{passwordValidated}$$

Substitution rule here refers to the get and set methods that replace new variables. The node that calls the *get* and *set* methods in AST is found, and new variables are generated by creating new methods, and a function is called to replace the

left and right sides of the get and set method expressions, and the value of the operator is kept unchanged.

6) Add Rule

$$Add\ Rule\ \frac{\phi}{Object}$$

Add Rule finds the node position of the variable or method call to be added, and after generating a new expression, insert it into the corresponding node position to complete the operation of adding.

7) Simplify For Loop Rule

$$Simplify\ For\ Loop\ Rule\ \frac{EnhancedForStatement}{ForStatement}$$

This rule refers to replacing the enhanced *for* loop with a generic *for* loop. Find the enhancement *for* loop and delete it, create a general *for* loop, and insert the new *for* loop into the node location where the original enhancement *loop* is located.

B. Method implementation

The implementation of simplified methods is based on the eclipse *JDT plug-in*, the specific method implementation is divided into the following steps:

1) Environment preparation: Since you are using the *JDT plug-in*, the first step is to install the *JDT plug-in* in Eclipse,the second is to configure an environment suitable for the *org.eclipse.jdt.core.dom** class, that is, to download the corresponding JAR package to use when the configuration program runs.

2) Parsing the AST: Firstly, use *ASTParser parsert = ASTParser.newParser(AST. JLS3)* statement to create a parser. Secondly, the Java source program to be parsed is generated as a string-typed parameter of the parser source code in AST. Finally, use the parser to create and return the AST context result CompilationUnit as the root node.

3) Modifying the AST:

a.Modify the data type: modify float into int.

b.Delete function implementation:delete variables, comments, statements,etc.

c.Substitution function implementation: Variable substitution *get* or *set* method.

d.Add function implementation: Add a variable or method call.

e.Simplified for loops: Rewrite the enhanced *for* loop as a general *for* loop.

4) AST convert into Java program: The file output stream (FileOutputStream) parses the modified string, and finally converts the string into a Java file and outputs it to the specified location.

III. EXPERIMENT

This section presents an example of a bank ATM withdrawal that shows how to simplify the source Java program by using AST and then verify the correctness with the KeY tool. Figure at github(https://github.com/1713022804/ATMexample) shows the simplification process of the bank ATM withdrawal, represented in AST, in which those on the left of the dashed line represents the original complex Java program, while the right side represents the simplified Java program.

Determine the format that Java programs verify in KeY, and then make specific simplifications according to the 7 rules provided in Section A of II. In this example, we mainly simplify the seven functions of the source Java program. The following are two examples showing how our rules are applied:

(1)In the *depositFunds* function, the Modify Data Type Rule is used to modify the parameter Float type into the Int type. The Delete Rule is applied to remove unnecessary comments and variables for verification of the method. Then the Substitution Rule is used to the *getPasswordValidated()* method, which is replaced by the PasswordValidated variable.

(2)In the *inputCard* function, the enhanced *For* Loop is simplified into the general *For* Loop by using the Simplify For Loop Rule, and Add Rule is applied to add variable C of BankCard data type to the method, then the Delete Rule is used to Delete comments, variables or statements in the methods. Finally, The *GetCardIDValidated()* method is replaced by the CardIDValidated variable, following the Substitution Rule.

IV. CONCLUSIONS

This paper presents seven rules you can use when simplifying your source Java programs into a format that the KeY can verify. Based on the AST, the simplified rules are derived, which are illustrated by using corresponding examples, and the simplified Java programs are verified based on the Java program-oriented verification tool KeY. In the future, we will continue to improve the simplification work based on AST and try to use empirical methods to evaluate our simplification rules.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (61862009).

REFERENCES

[1] S. Horwitz. Identifying the semantic and textual differences between two versions of a program. In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), pages 234–245, June 1990.

[2] G. Stoyle, M. Hicks, G. Bierman, P. Sewell, and I. Neamtiu. Mutatis Mutandis: Safe and flexible dynamic software updating. In Proceedings of the ACM SIGPLAN/SIGACT Conference on Principles of Programming Languages (POPL), pages 183–194, January 2005.

[3] Neamtiu I, Foster J S, Hicks M. Understanding source code evolution using abstract syntax tree matching[C]//Proceedings of the 2005 international workshop on Mining software repositories. 2005: 1-5.

[4] Ahrendt W, Beckert B, Bubel R, et al. Deductive Software Verification-The KeY Book[J]. Lecture notes in computer science, 2016, 10001.

[5] Ahrendt W, Bubel R. Functional verification of smart contracts via strong data integrity[C]//International Symposium on Leveraging Applications of Formal Methods. Springer, Cham, 2020: 9-24.

Trace4PF: A tool for Automated Decomposition of Problem Diagrams with Traceability

Yajun Deng†, Zhi Li†*, Hongbin Xiao†
†School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
Email: dengyajun@stu.gxun.edu.cn, zhili@gxnu.edu.cn, hongbinx1997@foxmail.com

***Abstract*—This paper provides a support tool for Jackson's Problem Frames approach – named *Trace4PF* for decomposing a global problem diagram into sub-problem diagrams. The tool provides a web browser interface with features such as drawing, editing and performing syntactical checking, and highlighting the trace of causal chain. A video demonstration of the tool is available at** *https://youtu.be/XSUVGGqEKkw*.

***Index Terms*—Cyber-Physical Systems, Problem Frames, Requirements Engineering, Decomposition and Traceability**

I. Introduction

Cyber-Physical Systems (CPS) play a crucial role in various fields and contain various elements (networks, sensors, actuators, displays, etc.) compared to the previous common systems [1]. Since CPS operate in an open, dynamic and diverse environment, their close interactions with the environment and users lead to many challenging problems. We argue, in this paper, that the most important challenge among these problems is a lack of an automated method or technique to reduce model complexity of such systems, and we provide a tool support for automatically decomposing a global problem diagram into sub-problem diagrams, once a traceability analysis is completed based on causal relationships elicited from stakeholders.

It is observed that Jackson's Problem Frames (PF) approach [2] is better suited for modeling, verifying and validating the complex contextual environments in relation to requirements of such systems. However, when faced with very complex CPS, the PF approach and existing tool support for modeling and analyzing the requirements in contexts usually encounter an overly large and global problem diagram, and fail to provide a traceability analysis based on causal reasoning, when decomposing the global problem into sub-problems.

This pape provides a support tool named *Trace4PF* for decomposing a global problem diagram into sub-problem diagrams, after syntactical checking, highlighting the trace of causal chains. The *Trace4PF* is a problem diagram tool that provides various editing and checking features for problem descriptions, which are prerequisites for problem decomposition with traceability analysis.

II. Tool Features

A. Problem Diagram Modeling

The GUI of *Trace4PF* is shown in Fig. 1. It is an online modeling tool for Problem Frames, in which stakeholders can draw their copies of problem diagrams using most mainstream web browsers. *Trace4PF* consists of three parts: (a) a canvas for the user to draw and edit; (b) predefined model elements for the user to draw and drop onto the canvas; (c) a toolbar that allows users to zoom the diagram to fit the current window, modify edge shape, upload or export files.

B. Syntactical Checking of Problem Diagrams

Once the diagram is completed by the user, a verification module is provided to help the user check the diagram for two types of syntactical errors.

Firstly, Syntax errors in labeling the phenomena and domain property. According to the grammars in PF [2], the format of the phenomena should be *DomainName!*{*phe1,phe2*}, so both *DomainName!*{*phe1phe2*} and *!*{*phe1,phe2*} are wrong, which must be corrected. The correct syntax of phenomena should follow the regular expression *regDP* below:

$$regDP = /\hat{}\texttt{DomainName!\{(}phe'*|phe\texttt{)*\}\$}/ \quad (1)$$

$$phe = /\hat{}\texttt{[a-zA-Z](\w)*\$}/ \quad (2)$$

$$phe' = /\hat{}phe\texttt{,\$}/ \quad (3)$$

In *regDP*, *DomainName* should be a name of a domain which is connected to the edge, and both *phe* and *phe'* are regular expressions as well. The domain property is defined using a regular expression named *regProp* as well as *regDP*:

$$regProp = /\hat{}\texttt{(}phePhe'*|phePhe\texttt{)*\$}/ \quad (4)$$

$$phePhe = /\hat{}phe\texttt{->}phe\texttt{\$}/ \quad (5)$$

$$phePhe' = /\hat{}phe\texttt{;\$}/ \quad (6)$$

If a domain or an edge has a syntax error, then the tool will show the domain or the edge and tell the user by highlighting the error in red.

Secondly, syntactical errors in circular causal relationships. In order to prevent circular causal relationships from reducing search efficiency. For example, in Fig. 1, phenomenon *pumpCmd* can indirectly evoke phenomenon *sugarUp*, and if *sugarUp* can directly or indirectly evoke *pumpCmd* at the same time (Suppose domain *Sensor* controls *pumpCmd* and *Sensor* has the property *sugarUp→pumpCmd*), then the tool would warn user that a loop exist in the diagram and show the loop.

*corresponding author: zhili@gxnu.edu.cn
DOI reference number: 10.18293/SEKE2022-181

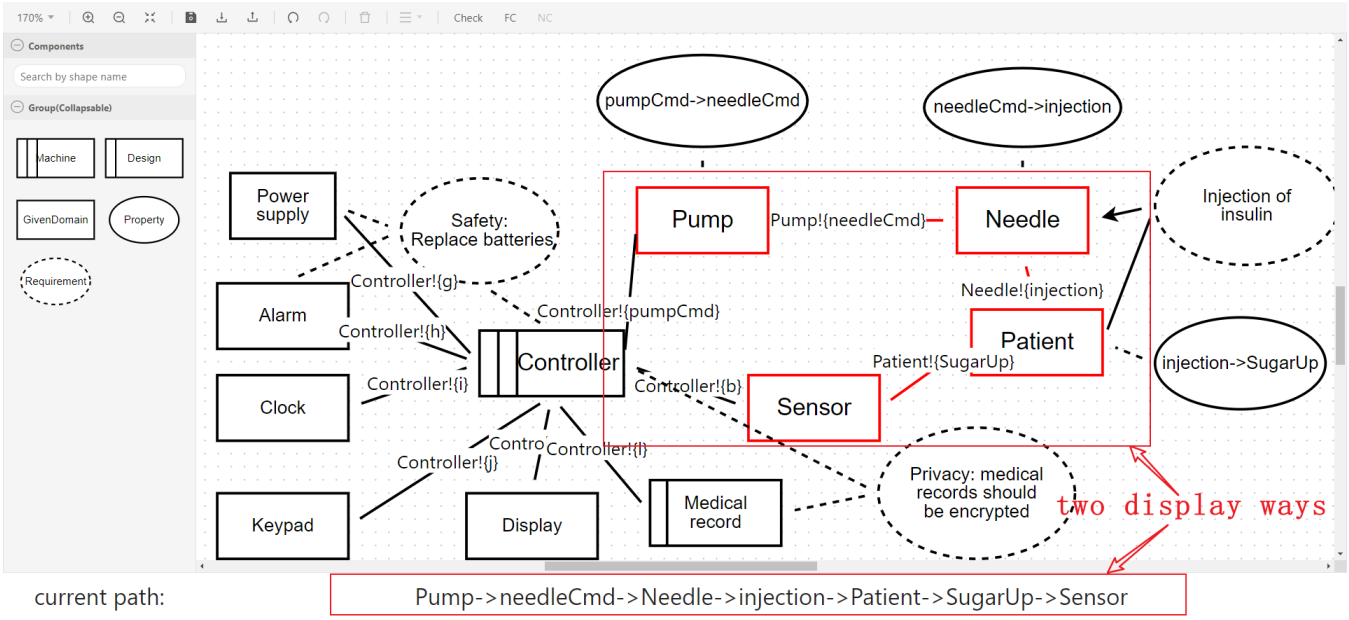


Fig. 1. The insulin control problem diagram modelled using our Trace4PF tool [3]

C. Decomposing Problem Diagrams with traceability

After the above syntactical checking is passed, the user can use and experience the core function of the tool – problem diagram decomposing with traceability. The current version of the tool provides the following three search options based on elicited causal relationships.

- Heuristic search. This method allows the tool user to explore all possible causal chains with traceable permissible paths (for reasons of space, this feature can be shown in the tool demo video).
- End-to-end search. This method shows all the permissible paths from a starting domain *Pump* to an ending domain *Sensor*, as shown in Fig. 1.
- Closed loop search. This method shows all the permissible paths starting from and ending at the Machine domain[1].

The tool can display all the permissible paths in both graphically and textually, as shown in Fig. 1.

III. Related Work

In recent years, a number of scholars have researched and developed problem framing tools. Some examples are as follows: Chen et. al. proposed DPtool [4], a tool to guide problem decomposition through scenario projection. Unlike our work, their tool has not yet implemented automated decomposition of problem diagrams with traceability. In this paper, we have extended the applicability of the current tool sets available to deal with more complex requirements inherent in CPS, with a web-based interface for ease of use.

[1]this is the only case where circular causal relationship is allowed starting from and ending at the *Machine* domain

IV. Conclusion

This paper presents the *Trace4PF* tool for automated decomposition of problem diagrams with traceability. The tool can be used through a web browser. The implementation of this tool uses AntV's open diagram editing engine X6 [2] – JavaScript Diagramming Library. In the future, we will continue developing this tool to provide more features, such as measuring the complexity of problem diagrams and automated test case generation.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (61862009), Guangxi "Bagui Scholar" Teams for Innovation and Research.

References

[1] Z. Jin, X. Chen, Z. Li, and Y. Yu, "RE4CPS: requirements engineering for cyber-physical systems," in *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*, D. E. Damian, A. Perini, and S. Lee, Eds. IEEE, 2019, pp. 496–497. [Online]. Available: https://doi.org/10.1109/RE.2019.00072

[2] M. Jackson, *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001.

[3] G. Liu, Z. Li, and Z. Ouyang, "CARE: A computer-aided requirements engineering tool for problem-oriented software development," in *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*, H. Xu, Ed. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2015, pp. 727–729.

[4] X. Chen, B. Yin, and Z. Jin, "Dptool: A tool for supporting the problem description and projection," in *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*. IEEE Computer Society, 2010, pp. 401–402. [Online]. Available: https://doi.org/10.1109/RE.2010.58

[2]https://x6.antv.vision/zh

Access-Pattern-Aware Personalized Buffer Management for Database Systems

Yigui Yuan, Zhaole Chu, Peiquan Jin, Shouhong Wan
[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China*
[2]*Key Lab. of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei, China*
jpq@ustc.edu.cn

***Abstract*—Buffer management is an essential technology for database management systems. Traditional buffer management employs an empirical approach based on access recency or frequency which fails to adapt to access-pattern changes in various database applications. In this paper, we present a new access-pattern-aware buffer manager called PBM (*Personalized Buffer Manager*), which can detect the access patterns for each database file and use a specific buffering policy for each database file. In particular, we propose a workload classifier to detect the access pattern of a database file. Then, we partition the buffer into various zones, set different sizes for each zone, and select the most suitable buffering scheme for each zone. With such a mechanism, each zone is responsible for caching a specific database file, and we can realize a personalized buffer manager for different database files, which can improve the buffer efficiency and reduce the page I/Os of the buffer manager. We compare PBM with three existing buffering algorithms, including LRU, LFU, and LeCaR, on two workloads, namely a regular workload and a shifting workload, which are composed of different access patterns. The results show that PBM outperforms the three competitors in terms of hit ratio and page I/Os. As a consequence, PBM achieves 1.66x, 2.03x, and 1.39x hit-ratio improvements compared to LRU, LFU, and LeCaR, respectively, on the regular workload. While on the shifting workload, PBM achieves 1.90x, 1.55x, and 1.49x higher hit ratios than LRU, LFU, and LeCaR, respectively.**

***Keywords*—Access pattern, Buffer management, Personalized buffer manager, Classification**

I. Introduction

Buffer management is a key module in database systems to improve query performance [1]. The optimal buffer manager can always maintain the pages that will be requested in the future in the buffer so that future requests can hit in the buffer. Generally, as the buffer size is usually limited, we need to use a replacement algorithm to evict some pages out of the buffer when the buffer is full. Therefore, most of the previous works on buffer management focus on the study of buffer replacement schemes, which highly determine the performance of buffer management in database management systems (DBMSs).

A buffer replacement policy aims to evict the most useless pages out of the buffer by predicting the future usage of pages. As future accesses are hard to be predicted, traditional DBMSs employ some empirical algorithms to perform buffer replacement. The most well-known algorithm is LRU (Least Recently Used) [1]. It assumes that the least recently used page is least likely to be requested in the future. Thus, it always selects the least recently used page for a replacement. In one word, traditional buffer replacement algorithms use an empirical way to select the victim for replacement. However, a critical problem of such a mechanism is that they cannot adapt to workload changes. As a result, they may perform well under some kinds of workloads but show poor performance under other workloads. For example, the LRU policy works well under the workloads with high time locality, but has poor performance when the workload involves periodical scans (known as the scan nonresistance problem of LRU). Although a few works studied the adaptivity of LRU, such as AD-LRU [2] and LeCaR [3], their performance relies on the empirical setting of parameters, which are still empirical solutions.

This paper proposes a new idea to improve the efficiency of buffer management in database systems. Differing from the traditional empirical approaches, we present an access-pattern-aware personalized buffer manager called PBM (*Personalized Buffer Manager*). The main contributions of PBM can be summarized as follows.

(1) PBM proposes to use multiple sub-buffers (called zones), each of which is responsible for caching a specific database file. Such a design enables us to use different buffering policies for different database files. As a result, PBM is equipped with multiple buffering policies rather than a single policy in traditional buffer management.

(2) PBM can detect the access pattern of each database file according to the access sequence. A workload classifier is proposed for the detection of the access pattern. Based on the detected pattern, PBM sets different zone sizes and enables different buffering policies for database files.

(3) We compare PBM with three existing buffering algorithms, including LRU, LFU, and LeCaR, on two workloads, namely a regular workload and a shifting workload, which are composed of five access patterns. The results show that PBM outperforms the three competitors in terms of hit ratio and page I/Os.

The remainder of the paper is structured as follows. Section II summarizes the related work. Section III details the structure and key technologies of PBM. Section IV reports the experimental results, and finally, Section V concludes the whole paper.

DOI reference number: 10.18293/SEKE2022-141

II. Related Work

In the past two decades, many well-known buffering policies have been proposed, e.g., LRU [1], LFU [1], ARC [4], 2Q [5], and LeCaR [3]. Most of these algorithms have been well explained in textbooks. The literature [1] presents a good survey on traditional buffer management policies.

LRU (Least Recently Used) [1] always evicts the least-recently-used page from an LRU queue used to organize the buffer pages, which are ordered by time of their last reference. It always selects as a victim the page found at the LRU position. The most important advantage of LRU is its constant runtime complexity. Furthermore, LRU is known for its good performance on workloads having high temporal locality. However, LRU does not exploit the frequency of references. Also, LRU is not scan-resistant.

LFU (Least Frequently Used) [1] removes the least frequently used page whenever the buffer is overflowed. The simplest method to employ an LFU algorithm is to assign a counter to every page that is loaded into the buffer. Each time a reference is made to that page, the counter is increased by one. When the buffer reaches the maximum capacity and a new page is waiting to be inserted, the buffer will search for the page with the lowest counter and remove it from the buffer.

ARC (Adaptive Replacement Cache) [4] is an adaptive caching algorithm that is designed to recognize both recency and frequency of access. ARC divides the cache into two LRU lists, T1 and T2. T1 holds items accessed once while T2 keeps items accessed more than once since admission. Since ARC uses an LRU list for T2, it is unable to capture the full frequency distribution of the workload and perform well for LFU-friendly workloads. For a scan workload, new items go through T1, protecting frequent items previously inserted into T2. However, for churn workloads, ARC's inability to distinguish between items that are equally important leads to continuous cache replacement.

The recent LeCaR algorithm [3] is an outstanding cache replacement algorithm that is based on reinforcement learning and regret minimization. The algorithm accepts a stream of requests for memory pages and decides which page to evict from a cache when a new item is to be stored in the cache following a "cache miss". LeCaR has been shown to be among the best performing cache replacement algorithms in practice [3]. Experiments have shown that it is competitive with the best cache replacement algorithms for large cache sizes and is significantly better than its nearest competitor for small cache sizes like ARC [4].

III. PBM: Personalized Buffer Manager

In this section, we detail the design of PBM. We first analyze the different access patterns and demonstrate that different buffering schemes are suitable for different access patterns, which motivates this study. Then, in Section III-B we present the workload classifier. Finally, in Section III-C we discuss the architecture and algorithms of PBM.

A. Analysis of Access Patterns

In the literature [6], the Turing Prize Winner, Michael Stonebraker, has summarized four types of common workloads: sequential accesses to blocks that are not seen nor re-visited, sequential accesses to blocks repeatedly visited, random accesses to blocks not seen nor re-visited, and random accesses to blocks where some blocks have a non-zero probability of reference. However, this definition of types is not quite fit for the purpose of choosing a replacement policy. For example, it makes no difference for LRU or LFU if the accesses are sequential or random. Therefore, in this paper, based on Stonebraker's definition, we define five access patterns, including *random*, *scan*, *skewed*, *cyclic*, and *vary*. The *random* pattern corresponds to the third type in Stonebraker's definition, where all pages in the file follow uniform distribution. The *scan* pattern corresponds to the first type. In the *skewed* pattern, some pages are accessed with a larger probability than the others, and in the *cyclic* patterns, a certain set of pages are cyclically accessed. The last pattern is a supplement to the former four patterns. It represents the access pattern where the hot region of the file varies over time.

We first generate workloads following the five access patterns and test the performance of several existing buffering policies on these access patterns. The results are shown in Fig. 1. We can see that no buffering scheme can maintain the highest hit ratio on all workloads. Also, different algorithms are suitable for different patterns. For example, LFU achieves the best hit ratio on the *cyclic* pattern but gets the worst performance on the *vary* workload. Therefore, a better way is to choose the best policy for a specific access pattern. Moreover, as the objects within a database may have different access patterns, it is better to use different policies for different database objects, which motivates the design of PBM.

B. Workload Classifier

We use two statistical features to distinguish the access pattern, namely *Correlation List* and *Page Coverage*. Both features are calculated based on the frequency histogram of the access. Let $H = \{a_1, a_2, ..., a_n\}$ be an access sequence, where a_k is an access, and $F = \{p_1, p_2, ..., p_m\}$ is the file it accesses. We first cut the sequence into segments, each with length S. Then, we calculate the frequency histogram for each segment. The frequency histogram is an m-length vector. The ith element of the vector is the frequency of page p_i in the segment. This vector conveys the page distribution information of the access. The reason why we segment the sequence is to recognize the change of distribution. Type A and B have stable distribution, while Type C's distribution varies.

(1) *Correlation List*. By comparing between segment histogram, we can figure out if the distribution has changed or not. This brings out our first feature: *Correlation List*. We use cosine correlation as the metric of the similarity between segments. Suppose the sequence has been cut into subsequences $\{s_1, s_2, ..., s_t\}$, where s_i is a segment, and $\{h_1, h_2, ..., h_t\}$ is the corresponding histogram list. Then the correlation list is $\{corr(h_1, h_2), corr(h_2, h_3), ..., corr(h_{n-1}, h_n)\}$. Figure 2

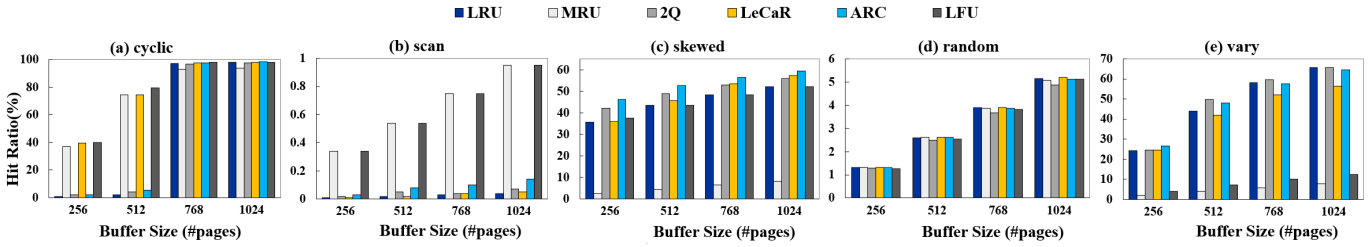


Fig. 1. The Performance Varying of Different Buffering Schemes on Various Access Patterns

shows the correlation list of the five access patterns. We can see that the correlation lists of *cyclic* and *skewed* are close to 1, showing that strong consistency exists between adjacent segments. Though the *random* and the *scan* pattern has a constant distribution, their correlation lists are close to or even below 0. For the *vary* pattern, the case is a little more complicated. Figure 2 shows when the segment length is 2000 and the phase changing period for *vary* is 5000, the correlation between two adjacent segments drops sharply every five segments.

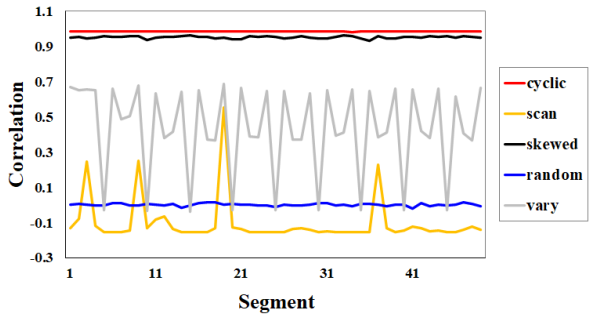


Fig. 2. The Correlation of the Five Access Patterns

(2) *Page Coverage*. For a segment of the workload, the *Page Coverage* is calculated as:

$$\frac{\#pages\ visited\ twice\ or\ more}{\#pages\ visited\ once\ or\ more}$$

The access in *vary* is concentrated on hot pages, while a *random* access is dispersed. Thus, there would be much more pages visited in *random* than in *vary*. However, most pages visited in *vary* will be visited twice or more. In *random*, on the contrary, only a small fraction of pages will be visited more than once. Suppose we have a file of size F, and two workloads with the *random* pattern and the *vary* pattern, respectively. The *vary* workload has a hot zone of size αF, and the probability of visiting its hot zone is β. If we pick a segment length larger than $2\alpha F$ but smaller than F, then the average visiting number of the pages in the hot zone of *vary* is:

$$\frac{average\ visiting\ number\ of\ the\ hot\ zone}{\#pages\ in\ the\ hot\ zone} \geq 2\beta,$$

while the average visiting number of the pages in *random* is:

$$\frac{\#visits\ in\ the\ segment}{\#pages\ in\ the\ file} \leq 1.$$

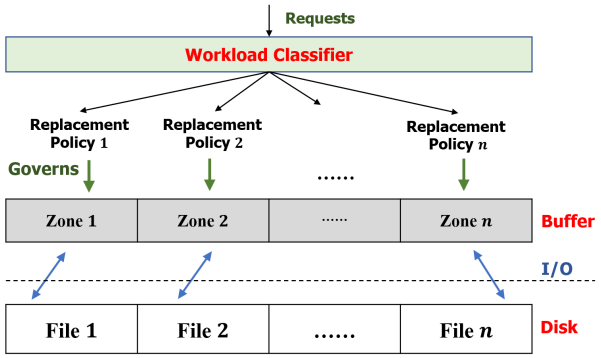


Fig. 3. The Structure of PBM.

As a result, the workload classifier based on *Correlation List* and *Page Coverage* is shown in Algorithm 1.

Algorithm 1: Workload Classifier

Input : *seg_len*: the segment length; ϵ: the threshold for correlation; γ: the threshold for page coverage; k: the threshold for number of correlations below the threshold; H: the access sequence to be classified;
Output: the type of H

```
1  segment_list = segment(H, seg_len);
2  page_coverage = the average page-coverage of all segments;
3  count = 0;
4  for s_i in segment_list do
5      if correlation(s_i, s_{i+1}) < ε then
6          count += 1;
7      end
8  end
9  if count > k then
10     if page_coverage > γ then
11         return C;
12     else
13         return A;
14     end
15 else
16     return B;
17 end
```

C. Architecture and Algorithms of PBM

Figure 3 shows the architecture of PBM. First, we divide the buffer into zones. Each zone is responsible for a database file in the disk and is governed by its own buffer replacement policy. The choice of a replacement policy depends on the

access pattern of the file. During the run time, the size allocated to each zone is variable. We group the five access patterns into three types. Type A includes *random* and *scan*. Type B consists of *cyclic* and *skewed*. Type C contains only the *vary* pattern. For a database file with Type A, since it will not contribute to the hit ratio under any policy, we shrink the size of its buffer zone. Thus, it will not pollute the buffer zone for other files. In our implementation, we make the zones for Type B managed by LFU and the Type C zones governed by ARC. The size of zones with Type B and C is dynamic. We use an evicting queue for each file to assess the buffer size it needs. The detailed PBM algorithm is as in Algorithm 2.

Algorithm 2: PBM

```
Input: B: the buffer; p: a page request;
1  adding_zone = B.find_zone(p);
2  if p ∉ adding_zone then
3      if p ∈ adding_zone.evict_list then
4          if adding_zone is not Type A then
5              adding_zone.evict_list.remove(p);
6              adjust_size = True;
7          end
8      end
9      if B is full then
10         if adding_zone is full then
11             adding_zone.evict();
12         else
13             evicting_zone = find_full_zone();
14             evicting_zone.evict();
15         end
16     end
17 end
18 adding_zone.request(p);
19 if adjust_size then
20     B.adjust_size(adding_zone);
21 end
```

Each zone with Type B or C has an evict list and a weight. The evict list keeps the metadata of the page evicted from the zone, and the total number of pages contained in the zone and its evict list is the total buffer size. At lines 11 and 14 of Algorithm 2, the *evict()* function evicts the page from the zone, records it at the head of the evict list, and evicts the tail element in the evict list if the total size exceeds the upper bound. When a missing page is in the evict list, we increase the weight of the zone, thus increasing its size. This is a better way to allocate space than only considering the file size or the number of visits. Because even though a file is large and more frequently accessed, the working set of it may be small, which means it requires less buffer space. But finding the missing page in the evict list shows that it is possible for the added space to be used for working set. Also, when we generate a new size allocation, we simply reset the size of each zone but do not make eviction immediately. When a miss occurs, and the buffer is full, we find a zone that is full or has overflowed to make eviction. The function $find_full_zone$ returns a zone with Type A first. The *AdjustSize* function invoked by Algorithm 2 is shown in Algorithm 3.

Algorithm 3: AdjustSize

```
Input: B: the buffer; f_k: the size of file k; F: the total size
       of the database; w_k: the weight of the zone k; S: the
       available size for zones of Type B and C; s_k: the size
       of zone k; i: the id of the adding_zone;
1  for every zone k with Type B or C do
2      w_k *= ∑w_j/(∑w_j + 1);
3  end
4  w_i += ∑w_k/(∑w_k + 1);
5  for every zone k of Type B or C and k≠i do
6      s_k = (w_k * S) / ∑w_k;
7  end
8  s_i = S - ∑s_k(k≠i);
```

IV. Performance Evaluation

In this section, we compare PBM to several replacement policies, including LRU, LFU, and LeCaR. We mainly focus on two metrics: hit ratios and page I/Os.

A. Setting

We run all experiments on a database consisting of ten files, each of which contains pages whose page number ranges from 0 to 100,000 (100k). Therefore, the database consists of 100k pages totally. To simulate the different access patterns of the files, we manually make each file have different access patterns. Each file contains 5k or 15k pages. Table I shows the details of each file used in the experiment. Note that each access pattern is associated with two files, with one file containing 15k pages and another file containing 5k pages. The default buffer size is set to 1,024 pages.

We generate 2500k page accesses for all the files. Further, we prepare two workloads based on the 2500k requests, namely a regular workload and a shifting workload.

(1) *Regular Workload*. In this workload, we distribute the page accesses to each file uniformly, i.e., each file receives 250k page requests. The requests to each file follow the access pattern of the file. For example, the 250k requests to file 1 satisfy the cyclic access pattern.

(2) *Shifting Workload*. In this workload, the page accesses to each file are skewed. To be more specific, we first participate the total 250k page requests into two parts, each of which contains 125k requests. Then, we let the 80% of the first half accesses focus on files 1 to 5 and the remaining 20% on files 6 to 10. Thus, in the first half requests, files 1 to 5 will be heavily accessed, but files 6 to 10 are not. For the second half 125k requests, we use the opposite setting, which is to make the 80% of the second half accesses focus on files 6 to 10 and the remaining 20% on files 1 to 5.

B. Performance on the Regular Workload

In this experiment, we compare the proposed PBM with existing buffering policies on the regular workload. Figure 4(a) shows the comparison of the hit ratios of PBM and other three existing buffering algorithms, including LRU, LFU, and LeCaR. Figure 4(b) shows the I/O comparison among all the

TABLE I
DESCRIPTION OF THE SYNTHETIC WORKLOAD.

	Page-ID Range	File Size (pages)	Access Pattern
File 1	0-15k	15k	cyclic
File 2	15k-20k	5k	cyclic
File 3	20k-35K	15k	scan
File 4	35k-40k	5k	scan
File 5	40k-55k	15k	skewed
File 6	55k-60k	5k	skewed
File 7	60k-75k	15k	random
File 8	75k-80k	5k	random
File 9	80k-95k	15k	vary
File 10	95k-100k	5k	vary

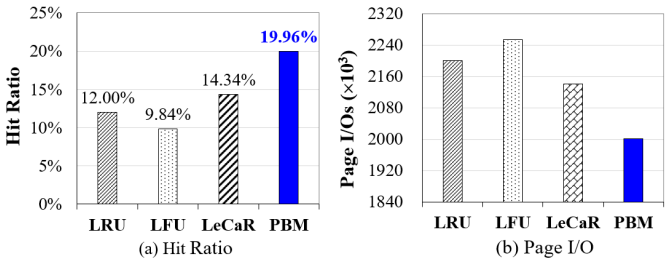


Fig. 4. Performance Comparison on the Regular Workload

compared buffering schemes. We can see that our proposed PBM achieves the highest hit ratio and the lowest page I/Os, owing to its dynamical algorithm selection according to access patterns. Particularly, PBM achieves 1.66x, 2.03x, and 1.39x hit-ratio improvements compared to LRU, LFU, and LeCaR, respectively. In addition, PBM reduces up to 11% page I/Os compared to its competitors. Note that LeCaR also shows good performance because it can adapt to access patterns. However, it only considers the recency and frequency of accesses and cannot detect other access patterns like *cyclic* and *vary*.

C. Performance on the Shifting Workload

In this experiment, we compare the proposed PBM with existing buffering policies on the shifting workload. Figure 5(a) shows the comparison of the hit ratios of PBM and other three buffering algorithms. Figure 5(b) shows the I/O comparison among all the compared buffering schemes. Compared with the experimental results on the regular workload, we can see that PBM achieves much more improvements over the three existing schemes. In particular, PBM achieves 1.90x, 1.55x, and 1.49x hit-ratio improvements compared to LRU, LFU,

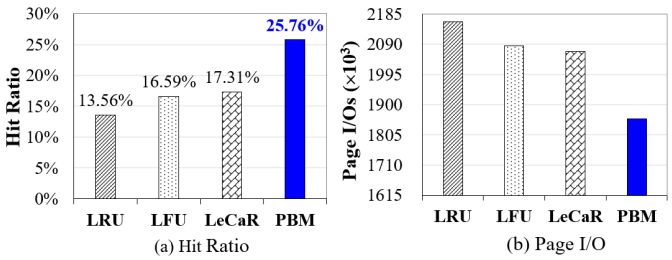


Fig. 5. Performance Comparison on the Shifting Workload

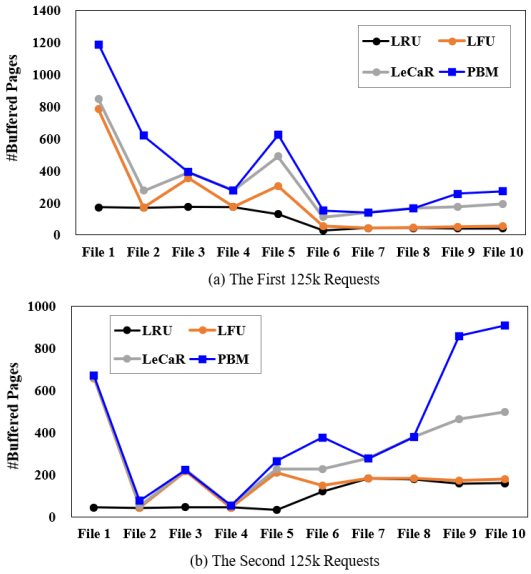


Fig. 6. The Change of the Buffered Pages When the Workload Shifts.

and LeCaR, respectively. Also, PBM reduces up to 14% page I/Os compared to its competitors. The higher performance of PBM on the shifting workload than on the regular workload is owing to its adaptivity, which can change the buffering scheme of each file according to the workload change.

To demonstrate the adaptivity of PBM more clearly, we calculate the average number of buffered pages for each file when performing the 250k page requests. As the first 125k requests in the shifting workload are focused on files 1 to 5, we expect that PBM can cache more pages of files 1 to 5 in the buffer. On the other hand, when running the second 125k requests that are focused on files 6 to 10, we expect that PBM can quickly adapt to the change of the access pattern and maintain more pages of files 6 to 10 in the buffer. As shown in Fig. 6, we can see that PBM can always keep more hot pages in the buffer when the workload changes with time.

D. Impact of the Segment Length

Segment length is a very important parameter for the classifier. The longer the segment is, the closer the histogram vector is to the distribution of the workload. However, a long segment can be bad for detecting the *vary* pattern, because when we calculate the histogram of several phases, the frequency of the hot zone is amortized, and the overall distribution resembles the *random* pattern. In this experiment, we test the influence of the segment length on the classifier. As shown in Fig 7, when we choose a long segment length, the gap between Type A and Type B is widened, meaning that the histogram of a long segment length can reflect the distribution better. In addition, a long segment length does not impede the separation of Type C, because both the *vary* pattern and the *random* pattern have a small correlation.

E. Comparison of Similarity Measures

The page classifier used in PBM employs the correlation-based approach. In this experiment, we consider other possible

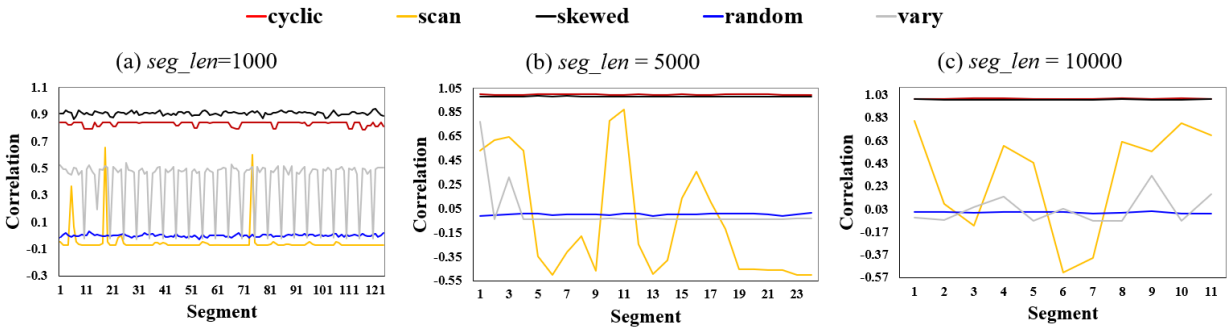


Fig. 7. The Impact of the Segment Length

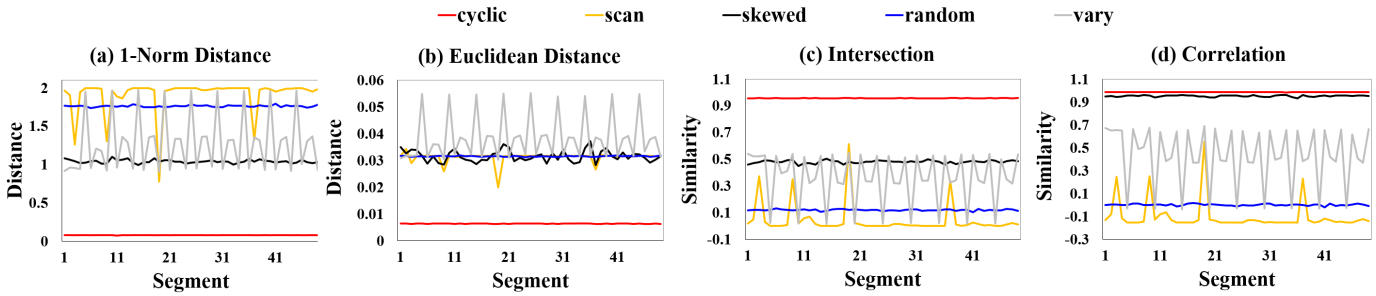


Fig. 8. Comparison of Different Similarity Measures

classifying metrics, aiming to show the superiority of the correlation-based classifier.

In addition to the correlation approach, we implement other three methods, which as listed as follows:

(1) 1-*Norm Distance*. This refers to the distance based on the 1-norm in a linear space.

(2) *Euclidean Distance*. This is the Euclidean distance between two vectors.

(3) *Intersection*. Given two vectors, the intersection between the vectors is defined as the sum of the smaller value for each element in the vectors, as shown in Equation 1.

$$Intersection(h_1, h_2) = \sum_{i}^{n} min(h_{1i}, h_{2i}) \quad (1)$$

Figure 8 shows the similarity of each metric when used for classifying the five access patterns. We can see that the 1-Norm Distance, Euclidean Distance, and Intersection all fail to classify the five access patterns clearly. Compared to the Correlation method, all the three methods cannot distinguish the *cyclic* from the other four patterns, which shows the superiority of the correlation approach proposed in PBM.

V. Conclusions

In this paper, we presented a new access-pattern-aware buffer manager called PBM (Personalized Buffer Manager). PBM can detect the access patterns for each database file and use different buffering policies for database files. We proposed a workload classifier to detect the access pattern and partitioned the buffer into different zones for different files. Each zone uses its own buffering policy for a database file, yielding a personalized buffer manager. We implemented PBM and compared it with three existing buffering algorithms on two workloads, including LRU, LFU, and LeCaR. The results suggested the efficiency of PBM.

In the future, we will investigate personalized buffer management for flash memory [7], [8] and use machine learning models to optimize the buffer management [9], [10].

Acknowledgements

This work was supported by the National Science Foundation of China (62072419). Peiquan Jin is the corresponding author.

References

[1] W. Effelsberg and T. Härder, "Principles of database buffer management," *ACM Transactions on Database Systems*, vol. 9, no. 4, pp. 560–595, 1984.

[2] P. Jin, Y. Ou, T. Härder, and Z. Li, "AD-LRU: an efficient buffer replacement algorithm for flash-based databases," *Data & Knowledge Engineering*, vol. 72, pp. 83–102, 2012.

[3] G. Vietri, L. V. Rodriguez, W. A. Martinez, S. Lyons, J. Liu, R. Rangaswami, M. Zhao, and G. Narasimhan, "Driving cache replacement with ML-based LeCaR," in *HotStorage*, 2018.

[4] N. Megiddo and D. S. Modha, "ARC: A self-tuning, low overhead replacement cache," in *FAST*, 2003.

[5] T. Johnson and D. E. Shasha, "2q: A low overhead high performance buffer management replacement algorithm," in *Proceedings of VLDB*, 1994, pp. 439–450.

[6] M. Stonebraker, "Operating system support for database management," *Communications of ACM*, vol. 24, no. 7, pp. 412–418, 1981.

[7] Y. Ou, T. Härder, and P. Jin, "CFDC: a flash-aware replacement policy for database buffer management," in *DaMoN*, 2009, pp. 15–20.

[8] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue, "CCF-LRU: a new buffer replacement algorithm for flash memory," *IEEE Trans. Consumer Electron.*, vol. 55, no. 3, pp. 1351–1359, 2009.

[9] S. Sethumurugan, J. Yin, and J. Sartori, "Designing a cost-effective cache replacement policy using machine learning," in *HPCA*, 2021, pp. 291–303.

[10] Y. Yuan and P. Jin, "Learned buffer management: a new frontier: work-in-progress," in *CODES/ISSS*, 2021, pp. 25–26.

DDMin versus QuickXplain – An Experimental Comparison of two Algorithms for Minimizing Collections

Oliver A. Tazl
Institute of Software Technology
Graz University of Technology, Austria
Graz, Austria
oliver.tazl@ist.tugraz.at

Christopher Tafeit
Institute of Software Technology
Graz University of Technology
Graz, Austria
christopher.tafeit@gmail.com

Franz Wotawa
Institute of Software Technology
Graz University of Technology
Graz, Austria
wotawa@ist.tugraz.at

Alexander Felfernig
Institute of Software Technology
Graz University of Technology
Graz, Austria
alexander.felfernig@ist.tugraz.at

***Abstract*—About two decades ago, two algorithms, i.e., DDMin and QuickXPlain, for minimizing collections, were independently proposed and gained attention in the two research areas of Software Engineering and Artificial Intelligence, respectively. Whereas DDMin was developed for reducing a given test case, QuickXPlain was intended to be used for obtaining minimal conflicts efficiently. In this paper, we compare the performance of both algorithms with respect to their capabilities of minimizing collections. We found out that one algorithm outperforms the other under given prerequisites and vice versa. These findings help to select the suitable algorithm for a given task.**

***Index Terms*—test case minimization, conflict minimization, software testing, application to diagnosis and configuration**

I. Introduction

There are many important tasks in different areas of Software Engineering (SE) and Artificial Intelligence (AI) requiring minimizing collections. In SE, localizing faults may require reducing inputs that lead to crashes or other unexpected behavior. For example, if a compiler crashes due to a given input program, fault localization becomes way more easy when knowing only those parts of the textual input that reveal the bug. In AI and there, for example, in Model-based Diagnosis (MBD) [1], [2] we rely on minimal conflicts used to compute minimal diagnoses. In any case, we have to deal with obtaining a – at least smaller – sub-collection that still fulfills the same criteria (or properties) as the original collection.

In SE and AI independently, two algorithms for minimizing a test case and a conflict were introduced about 20 years ago, substantially influencing in their respective research fields. In SE, Zeller and Hildebrandt [3] described the Delta Debugging algorithm DDMin and its use for simplifying failure-inducing inputs. In AI, Junker [4] suggested QuickXPlain for minimizing conflicts. Both algorithms rely on the general divide and conquer approach for minimization. Both algorithms aim at providing a smaller collection but come – at least partially – with limited guarantees regarding finding a smaller or even minimal solution. This is due to the fact that the computational complexity required would be exponential in the size of the collection.

Interestingly, these algorithms have not been considered in the respected research field of the other algorithm. Moreover – to our knowledge – nobody has ever compared these algorithms with respect to execution time and their capabilities of minimizing original collections. In this paper, we want to close this gap and provide an experimental evaluation where we compare DDMin with QuickXPlain on the same input collections and properties. The properties have been designed in a way allowing to make conclusions regarding in which cases one or the other algorithm behaves superior. Such an analysis has an impact to both SE as well as AI allowing to decide which algorithm to use under which circumstances. More concrete questions we want to answer in this paper are whether we can use DDMin for minimizing conflicts like QuickXPlain, or to use QuickXPlain for minimizing a test case instead of DDMin.

In summary, the content of this paper provides the following contributions: (i) it comes up with a framework allowing to compare DDMin and QuickXPlain directly (although they have been originally designed to fit different purposes), and (ii) it experimentally compares two different algorithms using a parametric set of inputs aiming at providing more insights regarding superiority of an algorithm in a particular application context.

We organize the remainder of the paper as follows: In Section II, we introduce the basic foundation of DDMin and QuickXPlain. Afterwards in Section III, we outline the underlying implementation and the experimental evaluation

DOI reference number: 10.18293/SEKE2022-172

procedure. Furthermore, we discuss the obtained evaluation results and derive some concluding remarks regarding the comparison between DDMin and QuickXPlain. Finally, we give an overview of related research and conclude the paper.

II. Basic foundations

To be self-contained, we outline the underlying foundations and the two algorithms DDMin and QuickXPlain. We start defining the underlying minimization problem to be solved. For this purpose, we assume that we have: (i) a collection of elements $C = \{e_1, \ldots, e_n\}$ where each element $e \in C$ is from a domain D, and (ii) a function $test$ that takes a collection of elements as input and returns either $\surd$ or $\times$, which is defined as follows:

$$test(x) = \begin{cases} \times & \textbf{if } x \text{ fulfills the given criteria or properties} \\ \surd & \textbf{otherwise} \end{cases}$$

Note that we assume $test(C)$ returns $\times$ for the original collection of elements. Furthermore, in the original definition of Zeller and Hildebrandt [3] $test(x)$, with $x \neq C$ may also return ? in case there is an unexpected behavior of x but which diverges from the behavior of C. However, in DDMin ? and $\surd$ are treated equivalently so there is no need to distinguish these two cases.

The problem of minimization of C with respect to a given $test$ function is to find an ideally smaller $C' \subseteq C$ (if it exists) for which $test(C') = \times$. If we want to have a really minimal C', we may come up with two corresponding definitions:

subset minimal: A collection $C' \subseteq C$ is called subset minimal if and only if there is no $C'' \subset C'$ where $test(C'') = \times$ holds. There might be more than one subset minimal solution. In the context of delta debugging [3] this type of minimum is referred to as local minimum.

cardinality minimal: Alternatively, a collection $C' \subseteq C$ is cardinality minimal if and only if there exists no smaller C'', i.e., $|C'| > |C''|$ where $test(C'')$ holds. In delta debugging, cardinality minimums are called global minimums.

Obviously, finding either subset minimal or cardinality minimal solutions is exponential in the size of the input collection C because we have to check all subsets of C. Hence, in practice, we may be more interested in finding a smaller solution if such a solution exists instead of a minimal one. It is worth noting in this context that DDMin only guarantees to return a solution with one element less, if such a solution exists. However, the evaluation indicates that in practice DDMin is way more efficient in removing unnecessary parts of a collection. Instead, QuickXPlain guarantees subset minimality but not cardinality minimality. Hence, when evaluating both algorithms, we are interested in how far away provided solutions are from the minimal one.

In the following, we describe the algorithms. Note that the used pseudo-code is adapted from the original one for allowing to use the collection C as well as the $test$ function as input directly. However, we did neither improve the algorithms nor change their originally stated behavior.

We first define helper functions that are used in the algorithms, namely *split* and *complement*.

split: *split* is able to divide a given collection of elements C into n parts leading to new collections $C_1 ... C_n$. Functions *split* returns collections that are pairwise disjoint ($C_1 \cap C_2 = \emptyset$), completely represent all elements of the original collection, i.e., $C_1 \cup C_2 ... \cup C_n = C$. Moreover, all the sub-collection have approximately the same size, i.e., For all i, and j: $|C_i| = |C_j| + x$ with $x \in \{0, 1\}$.

complement: The complement of a sub-collection C_1 is a collection comprising all elements of the original collection C that are not in C_1. I.e., *complement* is defined as follows: $complement(C_1) = \{x \mid x \in C \wedge x \notin C_1\}$.

In Algorithm 1, we depict the pseudo-code of the Delta Debugging (DD) algorithm DDMin of Zeller and Hildebrandt [3], which reduces the input collection using a divide and conquer strategy. The algorithm uses to some extent ideas from binary search for trilling down the failure inducing input systematically. The overall process implemented by DDMin has four steps: *reduce to subset*, *reduce to complement*, *increase granularity* and *done*. These steps split the input into smaller parts and combine them if needed to narrow down the faulty input. Correct parts were cut off to focus on the remaining faulty ones in order to produce a smaller input that triggers the faulty behaviour.

QuickXPlain (QXP) as shown in Algorithm 2, was introduced by [4] to solve over-constrained problems by providing explanations. Those are also calculated by a divide and conquer approach. The input is a problem instance which comprises an analysed set/collection ($\mathcal{A}$) and a background set/collection ($\mathcal{B}$). For our experiments we assume $\mathcal{B}$ to be empty. The function *test* is then used to determine the necessity of executing the algorithm. In the trivial case of a correct or not dividable input, the execution is stopped before entering the recursion. Next, the recursion is started. The procedure starts by partitioning the analyzed set into two, in our case equal-sized, subsets and analyze these subsets recursively until a minimal solution can be provided.

III. Implementation and Experimental Evaluation

In this section, we present the prerequisites behind the experimental evaluation in order to assure that results can be reproduced.

A. Prerequisites

The prerequisites comprise implementation details, the input collections used for the evaluation, and the execution environment. The objective behind the experimental evaluation was to answer the following 2 research questions:

RQ1 "*Does QuickXPlain behave superior compared to DDMin with respect to the execution time or vice-versa?*"

RQ2 "*Do both algorithms DDMin and QuickXPlain deliver minimal solutions?*"

Algorithm 1 DDMin [3]

Input: A collection c_f where $test(c_f) = \times$
Output: A potentially smaller sub-collection of c_f where $test$ returns $\times$.

procedure DDMIN(c_f)
  if $test(c_f) = \times$ **then**
    $ddmin2(c_f, 2)$
  end if
  return c_f
end procedure
procedure DDMIN2(c_f, n)
  if $|c_f| = 1$ **then**
    return
  end if

  $\Delta_1, ..., \Delta_n \leftarrow split(c_f, n)$

  while $i = 1...n$ **do** ▷ reduce to subset
    if $test(\Delta_i) = \times$ **then**
      $ddmin2(\Delta_i, 2)$
      return
    end if
  end while

  $\nabla_1, ..., \nabla_n \leftarrow complement(c_f, \Delta_1, ..., \Delta_n)$

  while $i = 1...n$ **do** ▷ reduce to complement
    if $test(\nabla_i) = \times$ **then**
      $ddmin2(\nabla_i, max(n-1, 2))$
      return
    end if
  end while

  if $n < |c_f|$ **then** ▷ increase granularity
    $ddmin2(c_f, min(|c_f|, 2 * n))$
    return
  end if

  return c_f ▷ done
end procedure

Algorithm 2 QuickXPlain [4], [5]

Input: a pair $\langle \mathcal{A}, \mathcal{B} \rangle$ where $\mathcal{A}$ is the analyzed set and $\mathcal{B}$ is the background, which we assume to be the empty set in the context of this paper.
Output: a minimal set wrt. $\langle \mathcal{A}, \mathcal{B} \rangle$, if existent; pass, otherwise

procedure QXP($\langle \mathcal{A}, \mathcal{B} \rangle$)
  if $test(\mathcal{A} \cup \mathcal{B}) = \surd$ **then**
    return pass
  else if $|\mathcal{A}| = \emptyset$ **then**
    return $\emptyset$
  else
    return QXP'($\mathcal{B}, \langle \mathcal{A}, \mathcal{B} \rangle$)
  end if
end procedure
procedure QXP´($\mathcal{C}, \langle \mathcal{A}, \mathcal{B} \rangle$)
  if $\mathcal{C} \neq \emptyset$ **and** $test(\mathcal{B}) = \times$ **then**
    return $\emptyset$
  end if
  if $|\mathcal{A}| = 1$ **then**
    return $\mathcal{A}$
  end if
  $\mathcal{A}_1, \mathcal{A}_2 \leftarrow split(\mathcal{A}, 2)$
  $X_2 \leftarrow$ QXP'($\mathcal{A}_1, \langle \mathcal{A}_2, B \cup \mathcal{A}_1 \rangle$)
  $X_1 \leftarrow$ QXP'($X_2, \langle \mathcal{A}_1, \mathcal{B} \cup X_2 \rangle$)
  return $X_1 \cup X_2$
end procedure

In order to answer these research questions we implemented both algorithms, i.e., DDMin and QuickXPlain, using the programming language Java 17 using libraries for logging data as a foundation. As stated we implemented the algorithms without manual optimisations for improving the execution time. Hence, we relied on the pseudo-code and description provided by the respective originators of the algorithms.

To implement the $test$ function required, we implemented a configurable test oracle, which uses JSON files to load and store input configurations. These configurations store the given collection of elements $e_1, \ldots, e_n$ of cardinality n, where each element e_i is marked as either neutral or failure-inducing element. The test oracle now implements the function $test$ as follows: The test oracle fails on a particular not necessarily strict subset C of the collection $e_1, \ldots, e_n$, if C contains all failure-inducing elements of in $e_1, \ldots, e_n$. In this case the test oracle returns $\times$ and otherwise $\surd$.

We use the configurations for coming up with different test inputs for DDMin and QuickXPlain. For this purpose, we created the configurations automatically using a program comprising the size n of the collection, and the number of failure-inducing elements (fail elements for short) k as inputs. For the experiments, we generated two types of inputs. The first type comprises tests where clusters of k fail elements arrange at the beginning of the collection, after k elements, after $2 \cdot k$ elements, etc. until reaching the end of the collection. In this type, which we refer as **cluster test input**, we only have one cluster of fail elements in each collection. In the second type of inputs, which we refer to as **random test input**, we generate collections of size n and randomly select k elements in this collection to be fail elements.

For the experiments, we created 3 sets of cluster test inputs of size 10,000 with 50, 500, and 1,000 fail elements respectively. For each of these sets we moved the cluster from the beginning to the end to obtain all cluster test inputs. For the random cluster with also relied on collections of size $n = 10,000$ considering 50, 500, and 1,000 randomly selected fail elements. For the random cluster, we generated 300 of such different configurations. In total, we executed about 400 different configurations. It is worth noting that we selected

collections of size 10,000 because of obtaining reasonable execution times ranging from milliseconds to less than 1 hour.

We published the implementation of the algorithms, the used configurations for the experimental analysis, the batch programs for running the experiments, as well as all data obtained is on Github to be used for further research.

B. Results

In order to answer the two research questions, we carried out the experiments, where we executed each input configuration ten times. Reported results are averaged to reduce the side effects of the execution environment on the results. In the following, we report execution time in milliseconds (ms) or seconds (s). The Java implementation of the algorithms was executed using the OpenJDK 17.0.1 Hotspot JVM using a computer with the following configuration: AMD Ryzen 9 3900x 12-Core 3.8 GHz processor and 64 GB RAM running Windows 11. Note also that the experimental evaluation is automated allowing to re-execute it on demand.

To answer **RQ1**, we measured the execution time required for minimizing the different input configurations. As already said we used two different types of configuration, i.e., the cluster test input and the random test input. In Figure 1, we display the result of the cluster test input where we combined a collection of 10,000 elements with 50, 100, and 1,000 fail elements respectively. All fail elements are in the same cluster, which we move from the left to the right of the collection for obtaining different input collections. Note when using a block of 1,000 fail elements, we obtain input collections, where the fail element cluster comprises elements 0-999, 1,000-1,999, etc.

The results for the cluster test input allow us to derive the following differences in the behavior of DDMin and QuickXPlain. First, DDMin comes with an almost constant execution time behavior with the exception of a cluster of fail elements in the middle. In this case, depending on the number of fail elements, the behavior of DDMin may even be worst when compared to QuickXPlain. This behavior might be due to the fact that DDMin requires to increase cardinality as well as to compute complements in order to find the cluster. QuickXPlain, however, has a more or less linear execution time behavior with respect to the position of the fail element cluster. Fail elements at the beginning (i.e., at the left side of the collection) lead to an execution time similarly to DDMin, whereas DDMin is superior when the fail element cluster is located on the right side of the collection.

In Figure 2, we summarize the results of the random test input configurations containing 10,000 elements and 50, 100 and 1,000 random distributed fail elements within. We see that QuickXPlain performs exceptionally well in this case compared to DDMin. In all cases, QuickXPlain outperforms DDMin.

Research question **RQ1** can now be answered as follows. QuickXPlain does not necessarily outperform DDMin when

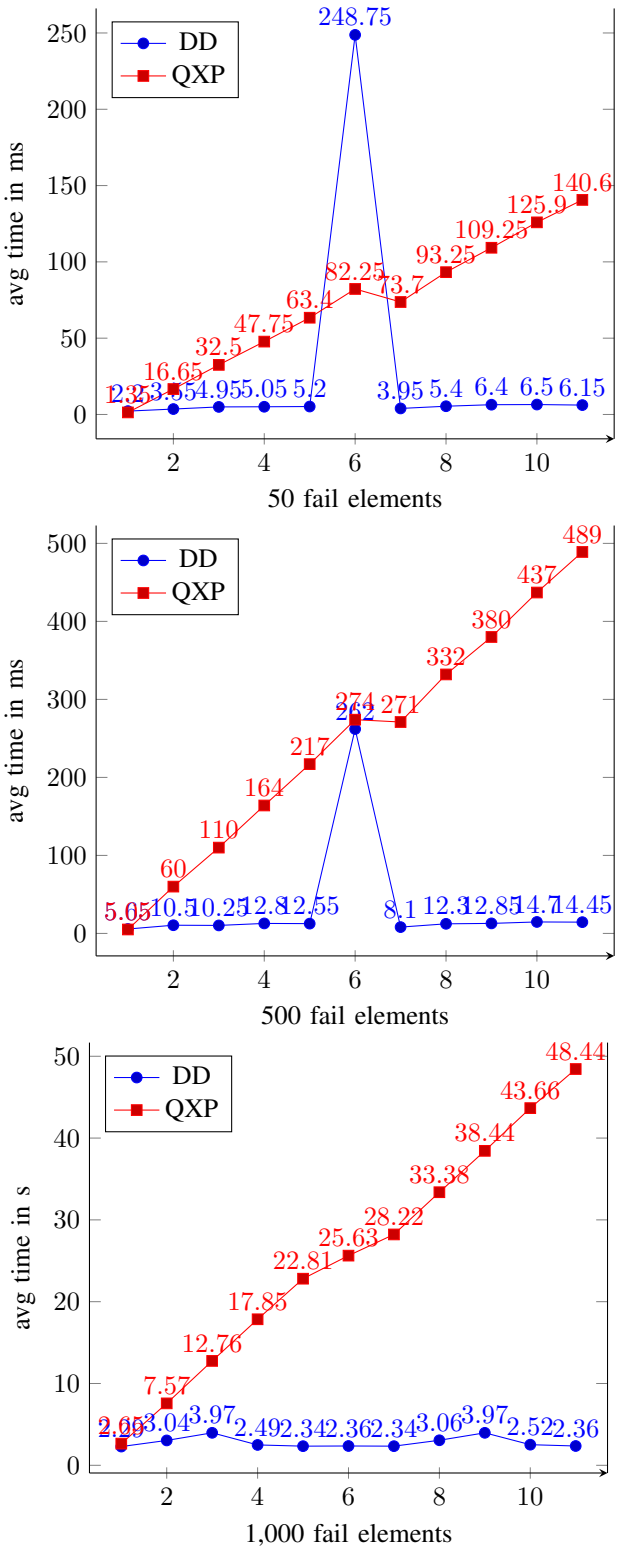


Fig. 1: Execution time results obtained using the cluster test inputs comprising 10,000 elements in combination with 50, 100 and 1000 fail elements respectively.

Link not given to fulfill given double-blind review requirements

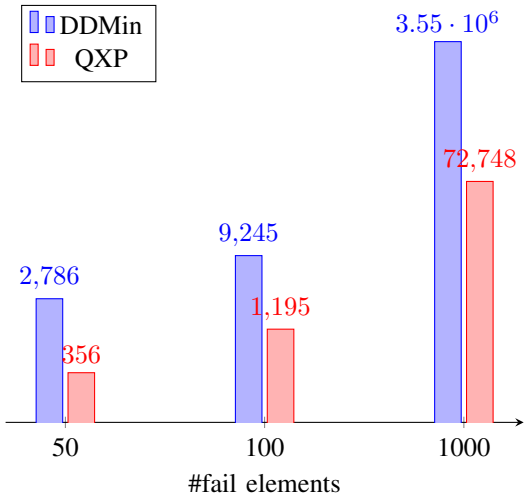


Fig. 2: Execution time results obtained using the random test inputs comprising 10,000 elements, and randomly selected 50, 100, and 1,000 fail elements. The depicted execution time is the average execution time in milliseconds.

considering execution time. In case of a single cluster of fail elements within a given collection DDMin seems to be superior, but execution time depends on the position of the cluster in the collection as well as on the number of the fail elements. If the number is larger, DDMin seems to be a better choice. However, in case of many clusters (like in the case of the random test input), QuickXPlain is faster than DDMin. Hence, depending on the given input arguments either QuickXPlain or DDMin should be chosen. For test case minimization (especially when considering inputs of a compiler), where there is most likely one cluster of interest, DDMin seems to be more appropriate. For conflict minimization, where there are many different randomly distributed small clusters, QuickXPlain seems to to be the algorithm of choice. However, further experimental evaluations are required considering real-world test inputs instead of synthetically generated example inputs as we used in our experimental evaluation.

To answer research question **RQ2**, we further compared the output, i.e., the minimized collection, of DDMin and QuickXPlain obtained for all the different test inputs with the optimal solution, i.e., the fail elements in the respective collection. For all test inputs, both algorithms returned the optimal solution as outcome. Hence, for the given test inputs, we can answer **RQ2** with yes. However, this result seems not to be conclusive and further experiments are required, considering real-world test inputs as well as more sophisticated synthetic examples.

C. Threads to Validity

Like for all experimental evaluations, there are different threats to internal and external validity, which we discuss. Regarding internal validity, we have to mention the computational environment comprising hardware and software used. This includes the operating systems as well as the as the programming language used. In particular, when relying on Java and its virtual machine, we know mechanisms like garbage collection and just in time optimization that we are not able to control, but influencing measured execution time. We try to mitigate those effects by repeating the tests and averaging the obtained time results. Moreover, it is worth mentioning that we implemented both algorithms in the same language making use of the same libraries not using any optimizations. Hence, we do not expect any bias in the measured execution time that originates from an implementation.

Regarding external validity, we have to mention that the evaluation is based on solely two different test input categories namely the cluster and the random test input. For the random case, the outcome was always identifying QuickXPlain as the fastest algorithm. For the cluster, DDMin can be said to be superior in most of the cases. Although, these results allow to state superiority on average in some more specific cases, such results may not be generally valid in all contexts including considering real-world example inputs. However, at least for those examples close to the synthetic one, we would expect a similar outcome.

IV. Related Research

The overall goal of this paper is to compare two algorithms that support the minimization of conflicts. Basic related minimality properties are *subset minimality* and *minimal cardinality* where the latter is more restrictive, i.e., also takes into account the criteria of subset minimality. Which criteria should be applied depends on the corresponding application context. Subset minimality is useful, for example, if a preference relationship can be defined over the given set of conflict candidates (e.g., given component failure probabilities or user preferences with regard to a set of product properties) [6], whereas minimal cardinality is useful in the case of non-available preference relationships (e.g., when searching failure-inducing inputs in the context of software testing) [3]. Especially in real-time scenarios, minimality criteria have to be relaxed to find a trade-off between conflict identification costs (time efforts) and costs for conflict resolution [7].

The algorithms discussed in this paper can be regarded as specific instances of so-called explanation algorithms [8]. DDMin [3] as well as QuickXPlain [4] support the determination of *minimal conflict sets* (fulfilling the criteria of subset minimality) which are also denoted as *minimal unsatisfiable subsets* (MUS) [9] or *minimal unsatisfiable cores* (MUC) [10]. Minimal conflict sets are well-suited for supporting the identification of minimizing collections in the context of test case minimization but as well in explorative interactive settings such as knowledge-based configuration [11] where users should be better supported in understanding relationships between different product properties.

In other scenarios, we are more interested in explanations that help to restore consistency, for example, [12], [6] focus on consistency restoration of inconsistent knowledge bases. In such scenarios, conflict sets are used as input for a hitting set algorithm [1] that helps to determine minimal diagnoses which are also denoted as *minimal correction subset* (MCS) [9]. In

contrast to hitting set based conflict resolution (diagnosis), direct diagnosis helps to determine hitting sets without the need of predetermining minimal conflicts sets. An example algorithm is FastDiag [13] which follows a divide-and-conquer based approach for identifying minimal hitting sets.

There is also a natural relationship between minimal conflict sets and minimal diagnoses in terms of a duality property [14]: for a given set CS of minimal conflicts we are able to determine a corresponding set DS of minimal diagnoses using a HSDAG based approach [1]. Vice-versa, we are able to derive exactly CS if we construct a HSDAG for DS.

Finally, the complement of a minimal hitting set, i.e., a minimal correction subset (MCS), is a so-called *maximal satisfiable subset* (MSS) [9]. Whereas MCSs Δ are characterized by the property that no subset of Δ fulfills the property that all conflicts can be resolved, MSSs Γ are characterized by the property that no extension of Γ remains satisfiable.

Summarizing, the two algorithms analyzed in this paper help to determine minimal conflicts sets which can then be exploited to determine corresponding minimal correction subsets as well as maximal satisfiable subsets.

In the context of software engineering and in particular testing, the minimization of collections is an important task. Besides minimizing a particular test case using Delta Debugging, the optimization of test suites (see, e.g., [15]) is of interest. There have been several algorithms proposed including Greedy algorithms [16] adopting solutions for the well-known set covering problem. In any of these cases, there is more information available than only a collection and a $test$ function. For test suite minimization, we usually know the influence of each test to the execution of a program, i.e., the statements that are executed. In the case of minimizing one test case, such knowledge is usually not available. Therefore, we focused solely on Delta Debugging for comparing it with QuickXPlain.

V. Conclusions

In this paper, we presented the outcome of an experimental evaluation of two algorithms for minimizing collections, i.e., QuickXPlain and DDMin, which originated from the different research areas of Artificial Intelligence and Software Engineering. In order to allow algorithm comparison, we provided a framework that takes a collection and a test function as input, and calls the algorithms for computing a sub-collection that still returns the same test function result. The underlying objective behind the research was to clarify whether one of the algorithms is superior with respect to execution time or the obtained minimization output.

To answer this question, we came up with different test examples generated automatically based on certain parameters. Based on our experiments, we were able to come up with the following results: (i) in cases where minimization has to deal with one cluster, i.e., a set of elements in the collection, which are in close proximity, DDMin provides an almost constant execution time behavior outperforming QuickXPlain. (ii) in case of random distribution of elements to be selected in a collection, QuickXPlain is superior with respect to its execution time. (iii) both algorithms always returned the smallest possible sub-collection. Hence, it seems that both algorithms were well designed for their particular area of use, i.e., minimizing a test case (DDMin), and minimizing conflicts (QuickXPlain). However, it is required to carry out further experiments considering real-world examples from the application domains of test case and conflict minimization, as well as more different generated examples for identifying additional parameters influencing the execution time behavior as well as the capabilities of finding a minimal solution.

Acknowledgement

The research was supported by ECSEL JU under the project H2020 826060 AI4DI - Artificial Intelligence for Digitising Industry. AI4DI is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2019 and April 2022. More information can be retrieved from https://iktderzukunft.at/en/ bmvit.

References

[1] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[2] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[3] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, feb 2002.

[4] U. Junker, "Quickxplain: Preferred explanations and relaxations for over-constrained problems," in *Proceedings of the 19th National Conference on Artifical Intelligence*, ser. AAAI'04. AAAI Press, 2004, p. 167–172.

[5] P. Rodler, "Understanding the quickxplain algorithm: Simple explanation and formal proof," *CoRR*, vol. abs/2001.01835, 2020. [Online]. Available: http://arxiv.org/abs/2001.01835

[6] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, "Consistency-based diagnosis of configuration knowledge bases," *Artificial Intelligence*, vol. 152, no. 2, pp. 213 – 234, 2004.

[7] A. Felfernig, R. Walter, J. Galindo, D. Benavides, M. Atas, S. Polat-Erdeniz, and S. Reiterer, "Anytime diagnosis for reconfiguration," *Journal of Intelligent Information Systems*, vol. 51, pp. 161–182, 2018.

[8] S. Gupta, B. Genc, and B. O'Sullivan, "Explanation in constraint satisfaction: A survey," in *13th International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021, pp. 4400–4407.

[9] M. H. Liffiton and K. A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *Journal of Automated Reasoning*, vol. 40, pp. 1–33, 2008.

[10] I. Lynce and J. P. M. Silva, "On computing minimum unsatisfiable cores," in *7th International Conference on Theory and Applications of Satisfiability Testing*, Vancouver, BC, Canada, 2004.

[11] U. Junker, "Configuration," in *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, 2006, pp. 837–873.

[12] R. Bakker, F. Dikker, F. Tempelman, and P. Wogmim, "Diagnosing and solving over-determined constraint satisfaction problems," in *IJCAI'93*. Morgan Kaufmann, 1993, pp. 276–281.

[13] A. Felfernig, M. Schubert, and C. Zehentner, "An efficient diagnosis algorithm for inconsistent constraint sets," *AI for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, vol. 26, no. 1, pp. 53–62, 2012.

[14] R. Stern, M. Kalech, A. Feldman, and G. Provan, "Exploring the duality in conflict-directed model-based diagnosis," in *AAAI'12*. AAAI, 2012, pp. 828–834.

[15] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 649–678, 2011.

[16] L. Zheng, M. Harman, and R. Hierons, "Search algorithms for regression test case prioritization," *IEEE Trans. Softw. Eng.*, vol. 33, no. 4, p. 225–237, Apr. 2007. [Online]. Available: https://doi.org/10.1109/TSE.2007.38

Supporting Data Selection for Decision Support Systems: Towards a Decision-Making Data Value Taxonomy

1st Mathieu Lega
Namur Digital Institute (NaDI)
University of Namur
Namur, Belgium
0000-0003-1682-4920

2nd Christian Colot
Namur Digital Institute (NaDI)
University of Namur
Namur, Belgium
0000-0002-0520-7364

3rd Corentin Burnay
Namur Digital Institute (NaDI)
University of Namur
Namur, Belgium
0000-0002-0325-1732

4th Isabelle Linden
Namur Digital Institute (NaDI)
University of Namur
Namur, Belgium
0000-0001-8034-1857

***Abstract*—Data are ubiquitous and generate unprecedented opportunities to help making decisions within organizations. This has led so-called data-driven Decision Support Systems (DSS) to become critical, if not vital, systems for most companies. The design of such DSS raises important methodological challenges, since data-driven DSS should expose only useful information to decision makers, but data available in a company's database are numerous and not equally supportive. Failing to provide the right data to the right decision-maker may reduce the usefulness of a DSS, and can lead to lower quality decision outputs. This is particularly striking in the case of Self-Service Business Intelligence (SSBI) where users build DSS outputs themselves. In this paper, we elaborate on this idea of data profusion and propose a data selection criterion, namely the decision-making data value. To do this, we discuss the concept of value and its application to data and decision making, we review existing literature and propose a taxonomy of the dimensions of data value in the context of decision making. We also validate this taxonomy with semi-direct interviews and discuss the future research we plan to conduct as a way to apply this approach for the specification of high-value data-driven DSS.**

***Index Terms*—Data value, Data quality, Data utility, Decision making, Data selection, Decision Support System**

I. Introduction

The profusion of data available in organizations is clearly established and objectified. In 2020, an estimated 47 zettabytes (a thousand billions of gigabytes) of data has been produced all over the world [24]. By contrast, in 2010, that amount was only about 2 zettabytes. This exponential growth is expected to continue in the next 15 years to reach 2000 zettabytes in 2035 [24]. This inflation of accessible data has numerous – positive and negative – effects on many activities within an organisation [3]. This includes the process of supporting decision making, on which we focus our attention in this paper.

Every modern organisation likely generates significant quantity of data, needs to support decision makers who face more and more complex decision settings and dynamic environments, yet often struggles to provide them with timely and relevant decision-making support and notably data reporting. In practice, it is not uncommon for decision makers to face data sets that are too complex to be helpful, that have too many quality problems (missing values, encoding errors, etc.) to be useful, that are too isolated from other piece of data to generate real insight, etc. [11]. The purpose of so-called data-driven Decision Support Systems (DSS in the rest of this paper) is to mitigate those risks and provide smooth decision support to members of a company [22].

The proposition developed in this paper emerges in the context of Self-Service Business Intelligence (SSBI), a type of DSS where end-users have to select the pieces of data and visuals by themselves [1]. SSBI empowers end-users with the responsibility to produce dashboards by themselves, thereby reducing the time-to-delivery while improving alignment with business requirements. The question of selecting pieces of data to expose to the business users is central in DSS, and becomes even more critical in the case of SSBI [2]. Business-users indeed have little technical knowledge and little to no understanding of databases underlying a business application. As a result, the need to provide guidance and to help business users find what they really need in order to produce their own dashboards is significant.

This guidance is even more important considering three risks related to SSBI [17]; (i) dashboards may be overloading due to the presence of too much data, (ii) sub-optimal selection criteria may be applied to data by users, which may result in the omission of useful pieces of data or the inclusion of

This work was initiated in collaboration with the company Loth-Info and is partially financed by the Fonds Spécial de Recherche (FSR).
DOI: 10.18293/SEKE2022-104

irrelevant ones and (iii) bad quality data may be incorporated in the dashboard, resulting in reduced insights. All these risks involve spending resource and time on the implementation of unsupportive dashboards.

To address this problem, there is a need to find a criterion to select the most important pieces of data in the best possible way, in order to facilitate the process of designing SSBI dashboards and more globally DSS. This selection criterion should be understandable by business-users (it should not be too technical) yet incorporate important technical aspects whenever those aspects impact decision support. To the best of our knowledge, such data selection criterion has never been formalized in the literature on information management. As an answer, we advance our definition of a Decision-Making Data Value, or simply DMDV, criterion. This raises the following research questions:

1) What is *DMDV* in the context of data-driven Decision Support Systems and SSBI?
2) What are the different *dimensions* that compose our *DMDV* criterion?

In the remainder of this paper, we expand on these two questions. In section II, we position the concept of DMDV in the literature on software engineering in general. section III details our methodology to develop a taxonomy of relevant, distinct, measurable and comprehensive dimensions influencing DMDV. In section IV, we present this DMDV taxonomy. section V presents the future works. We conclude and present the limitations of this work in section VI.

II. Related Works

In this section, we review papers around three main topics: the concept of value in general, data quality and data value. We highlight some important observations to guide the reader.

A. Concept of Value

Before discussing the notion *data value*, it seems important to clarify our understanding of the notion of *value* since different definitions may be used depending on the situation [8], [20]. In [20], four main value situations are presented; (i) the exchange value that represents the amount of money needed to get a product, (ii) the esteem value that represents the price a customer is ready to pay for prestige or appearance, (iii) the use value that measures the functionalities of a product and (iv) the other value situations that group more particular situations such as the aesthetic value, the judicial value, the moral value or the religious value.

These definitions help us clarify what is meant by DMDV in the context of SSBI. Data is by definition a product, i.e. something that is produced. As a reminder, we focus in this paper on the value of data in the context of decision making within a company. We thus exclude operating or acquisition costs; for instance, we do not take into account the cost of purchasing the data from an external provider, neither do we account for the selling price of a data item. Similarly, we do not include in our DMDV the cost of collecting and recording the data, of maintaining a database or of using the data in a SSBI solution. Our goal is to support the selection of already available data for reporting purposes, not to help in determining if the use of a given piece of data is profitable for the company, i.e. if it generates more revenues than it generates costs. Our conceptualisation of value is thus not an exchange value. Neither the concept of "esteem value", nor the value situations (aesthetic, judicial, moral and religious) do relate to data aimed for internal decision making. However, the remaining concept (the use value) seems to be adequate to describe DMDV.

The concept of use value has been defined by Karl Marx in [19] when speaking about commodities. As defined by Marx, a commodity is "an object outside us, a thing that by its properties satisfies human wants of some sort or another". Based on this, he defines the use value of a commodity (citing [18]) as the "fitness to supply the necessities, or serve the conveniencies of human life". In [8], the use value is defined as "the specific qualities of the product perceived by customers in relation to their needs".

Based on these definitions, we can say that the decision-making value of data is a use value. Indeed, data may be seen as a thing that satisfies human wants, i.e. help for decision making. This brings us to define the decision-making value of a data item as: *its fitness to help decision making*. This brings us to the following observation:

> *Observation 1* - A data item is valuable if its use allows the decision maker to improve its decision-making.

B. Concept of Data Quality

The quality of a product or service likely influences its value. It is a commonsense observation in our daily lives, and this also applies in the specific context of data. Data quality has been discussed in the literature for a long time now, with many researchers trying to understand what data quality really represents and how it can be decomposed and/or measured. We can distinguish two important topics of research about data quality.

A first line of research is the *identification of the dimensions* that should be taken into account to decompose quality of data [6], [13], [15], [21], [26]. This issue has been found to be quite challenging and no standard emerged. A review of a large number of propositions has been conducted by [6]. It emphasized that some dimensions were more discussed than others in the literature, namely: accuracy, completeness, consistency, timeliness and currency. Moreover, the number of dimensions taken into account by authors varies quite a lot. While some authors derive only 5 dimensions from data quality, others decompose it into more than 15 dimensions.

> *Observation 2* - There is no strong agreement on the dimensions that influence the quality of data.

Another line of research *proposes a set of metrics* that may be used in order to assess the quality of data [6], [13], [15], i.e., to operationalize data quality. Various propositions have been made but no standard emerged. In [6], the authors also

include this second aspect in their review of several papers about data quality. While most dimensions have different propositions of metrics, some of these propositions are more used than others [6]. For example, accuracy, completeness and consistency all have one specific proposition of metric that is significantly more discussed than the other propositions for the same dimension. This is not the case for timeliness and currency.

Observation 3 - Some dimensions have metrics suggested in the literature, others not.

C. Concept of Data Value

The concept of data value is more recent in the literature than data quality. The works presented below typically address data value in different ways.

In [12], the authors define the concept of intrinsic value of data to support data quality assessment. Based on the idea that quality assessment must be contextual, this value is measured starting from the records of the data. Each record receives a specific value depending on its frequency of use. More concretely, most accessed records receive a high value while less used records get a low value. These records values are then translated into weights to compute data quality metrics such as completeness or accuracy. The idea is that all records do not deserve the same attention from a data quality point of view. Most used records are more critical than less used records to assess data quality for a specific organisation. This approach offers a first step to take into account the value of the data. However, it does not discriminate this value across features of the same dataset. Hence, this approach might lead to focus on curating potential redundant and noisy features if their data quality is low for frequently used records. Another work [23] considers data value to optimize data management. For this purpose, the authors define the concept of disparity of data value and suggest ways to measure it. The value of a record in a dataset is attributed based on the context.

Observation 4 - Data value is context-dependent.

A specific line of research proposes general methodologies to derive the value of data. These methodologies can be based on human input [3], [10], [16], data processing [4], [7] or both [5]. This ability to materialize the value of data in concrete numbers is all the more important as it is a crucial ingredient for a sound data governance and more generally for any decision supported by data [3], [9]. What stands out in the literature is the fact that there is no unique and non overlapping definition of the constituents of data value [3], [16]. However, based on these methodologies, four observations may be derived for the selection of the constituents of data value:

Observation 5 - The selected dimensions should impact data value in the context of use [3].

Observation 6 - Redundancy among dimensions should be minimized [6].

Observation 7 - The selected dimensions should be measurable [3].

Observation 8 - The selected dimensions should take a maximum of data value aspects into account [7].

III. Methodology

Our DMDV taxonomy was built on a 4-steps methodology.

The first step consisted in gathering the dimensions of data value already identified in the literature. The aim here was not to conduct a systematic literature review but to gather the most discussed dimensions of data value. We searched for articles on the search engine Scopus using the following query (”data value dimensions” OR ”data value assessment”) with the search being performed on the title, abstract and keywords of the articles. We also included the literature review of [6] on the dimensions of data quality to the analyzed articles because data quality is highly related to data value. We then extracted the dimensions of the retrieved articles.

The aim of the second step was to create a taxonomy of all the retrieved data value dimensions that are applicable to the context of databases. In order to realize this, we first dropped the dimensions that were out of this scope. Then, we used an open card sorting approach performed by the 4 authors of the paper [28]. We shuffled all the retrieved dimensions randomly and each author classified the dimensions individually. We then discussed our results together to obtain a final data value taxonomy. The card sorting was performed by the authors because it required to understand each dimension in the context of data value and thus to have knowledge about the literature.

The third step was the selection of the dimensions to include in our DMDV taxonomy based on the global data value taxonomy. The observations 5, 6, 7 and 8 that have been identified in the previous section allowed us to guide this selection. In the remaining of this article, we respectively refer to these observations as: relevance criterion, distinctiveness criterion, measurability criterion and comprehensiveness criterion. First, the lowest-level classes of the data value taxonomy were screened for their relevance for decision making. Then, for each selected class, we only kept the dimensions that are measurable, i.e. the dimensions that already have measures proposed in the literature. Finally, we applied our comprehensiveness criterion by selecting the dimension that was assessed by the four authors as the most representative for each class based on a discussion and a vote. In other words, we kept the dimension that represents the broadest concept. This allowed us to select a global dimension that takes into account the information of the whole class. In this methodology, 3 of our criteria are explicitly used i.e. relevance, measurability and comprehensiveness, while the remaining criterion is implicit. Indeed, the distinctiveness criterion is also part of the method as we selected only one dimension by class. It is important to note that the comprehensiveness criterion is also taken into account at the beginning of the process as we started from all the dimensions retrieved from the literature.

In the final step, we validated our DMDV taxonomy with data experts, from both the scientific and the business worlds. The criterion to choose them was that they must have at least 5 years of experience in the field of data processing and notably reporting. We presented them our taxonomy during semi-

structured interviews and asked them to review this taxonomy according to three axes: (i) the identification of potential missing or superfluous dimensions; (ii) the identification of potential missing or superfluous classes and (iii) the identification of potential misclassified dimensions.

IV. Results

We now detail the application of our methodology to derive and validate a DMDV taxonomy from the literature.

A. Identification of the dimensions of data value

The dimensions retrieved in step 1 are presented in table I. An expected observation is that some dimensions are not applicable to the context of a database. For example, site access is a dimension directly related to the context of data available on a website and is thus not applicable to a database. We thus dropped these dimensions for step 2.

B. Data value taxonomy

The result of step 2 is presented in figures 1 and 2. The dimensions with an asterisk next to their name have a measure proposed in the reviewed literature (this will be used in the next subsection). We separate the data value taxonomy in two figures for clarity purpose. Our data value taxonomy classifies the retrieved dimensions in two main classes: data quality and data utility. Data quality encompasses the dimensions about the data itself and the way it is encoded (such as completeness or correctness) and is divided into 4 subclasses: completeness, correctness, technical aspect and time aspect. Data utility encompasses the dimensions about the use of the data (such as ease of operation and relevance) and is divided into 5 subclasses: ease of use, legal aspect, monetary aspect, uniqueness and usability. For clarification purposes, the subclass "ease of use" encompasses the dimensions about the extent to which data may be used in an easy manner while the subclass "usability" is about the goals that may be achieved with the data.

C. DMDV taxonomy

Our step 3 was applied in turn to data quality and data utility. The 4 subclasses of data quality dimensions displayed in figure 1 were thus assessed for their potential impact on decision making. As the subclass "technical aspects" does not directly relate to this purpose, it was consequently rejected. Then, following our step 2, only the dimensions having measures proposed in the literature were kept. Finally, the broadest dimension was selected for each subclass, leading to the selection of the 3 following dimensions, that we define in the context of decision-making: (i) Completeness: the extent to which the available data is complete, are there enough values or is the data empty? (ii) Correctness: the extent to which the available data contains errors, can we believe what is encoded? (iii) Timeliness: the extent to which the available data is up-to-date, are the values still valid?

Turning now to the utility factor, the relevance of subclasses for internal decision making was checked. Monetary aspect and legal aspect do no contribute to this goal and they were accordingly discarded. Indeed, these 2 subclasses impact the ease to get and to use data but, in this work, we focus on data already available and usable for the organisation. For the 3 remaining subclasses, we kept the following dimensions that are assessed as measurable by the literature and having the broadest scope of their subclass, and we define them in the context of decision-making: (i) Interpretability: the extent to which the available data may be interpreted, do we understand what is encoded? (ii) Uniqueness: the extent to which the information embedded in the available data is unique, as several data items may contain the same information, do we already possess this information? (iii) Usability: the extent to which the available data contains useful information for decision making (this takes into account the current usage of data and the future objectives that could impact how this data is used), is the data used for decision making?

Integrating the results of this double application of the dimension selection process on both data quality and data utility, our DMDV taxonomy is presented in figure 3.

D. Validation of the DMDV taxonomy

In order to validate our DMDV taxonomy, we realized interviews until we reached a saturation threshold in the answers. This led us to conduct 7 interviews of both researchers and practitioners (3 researchers in the field of information management, 3 IT consultants and 1 data manager).

For the first axis, "identification of potential missing or superfluous dimensions", the respondents identified 3 potential missing dimensions: granularity (cited four times), the ability to be visualized (cited three times) and quantity (cited one time). One respondent also suggested a division of interpretability into format and meaning. Granularity and the ability to be visualized are in fact encompassed respectively in our dimensions usability and interpretability. Indeed, the granularity of the data directly impacts its possible usages and the ease to visualize the data impacts its interpretability. We thus revise our definitions of usability and interpretability to better express these aspects. Usability is "the extent to which the available data contains useful information and has the right level of granularity to be used for decision making". Interpretability is "the extent to which the available data may be interpreted and notably visualized". Quantity is not identified in the literature as a data value dimension and we argue that it is more an element that determines the need to find a data selection criterion than a DMDV dimension. We also argue that interpretability does not need to be divided at this point due to our comprehensiveness criterion and that this division should be kept in mind for the eventual design of an interpretability metric. Our dimensions are thus validated.

The second axis, "identification of potential missing or superfluous classes", only generated one comment as all the respondents except one completely agreed with the classes quality and utility. One participant suggested that usability could be a third class between quality and utility based on the cognitive process she follows when designing a dashboard.

TABLE I
RETRIEVED DIMENSIONS OF DATA VALUE IN THE LITERATURE

Source	Dimensions
Batini et al. [6]	accuracy, completeness, consistency, timeliness, currency, volatility, uniqueness, appropriate amount of data, accessibility, credibility, interpretability, usability, derivation integrity, conciseness, maintainability, applicability, convenience, speed, comprehensiveness, clarity, traceability, security, correctness, objectivity, relevance, reputation, ease of operation, interactivity
Brennan et al. [9]	usage, cost, quality, intrinsic, IT operations, contextual, utility
Brennan et al. [10]	operational impact/utility, dataset replacement costs, competitive advantage, regulatory risk, timeliness
Wang et al. [27]	content, credibility, critical thinking, copyright, citation, continuity, censorship, connectivity, comparability, context, site access and availability, resource identification and documentation, author identity, author authority, information structure and design, content relevance and scope, content effectiveness, accuracy and balance of content, navigation within documents, link quality, aesthetic and emotional aspects, information source, scope, discussion, technology factors, text format, information organization, price, availability, user support system, authority, credibility, accuracy, reasonableness, support, timeliness, integrity, consistency, acquisition cost
Attard et al. [5]	usage, quality, data, infrastructure
Holst et al. [14]	usage, quality, monetization, data sourcing costs, data processing and analysis needs, importance for business model and decisions
Stein et al. [25]	usage, quality, costs, completeness, conciseness, relevance, correctness, reliability, accuracy, precision, granularity, currency, timeliness
Bendechache et al. [7]	volume, usage, utility, replacement cost, legislative risk, timeliness, competitive advantage, quality, security

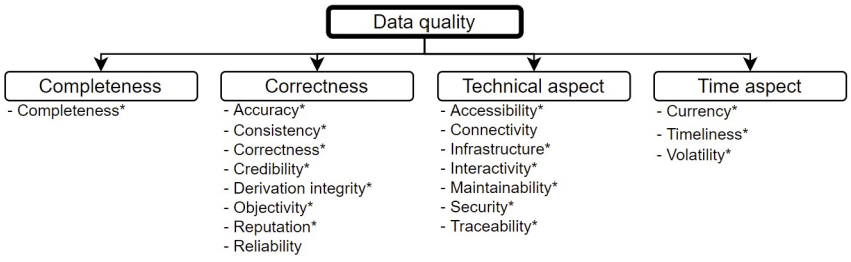


Fig. 1. Data quality part of the data value taxonomy

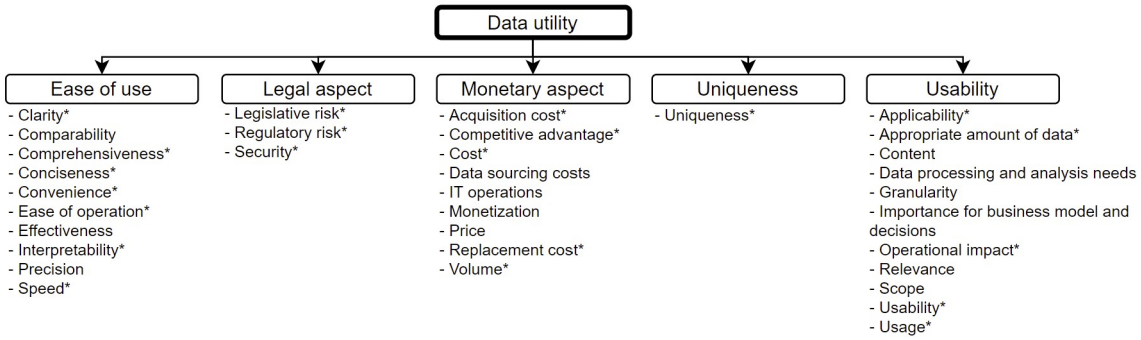


Fig. 2. Data utility part of the data value taxonomy

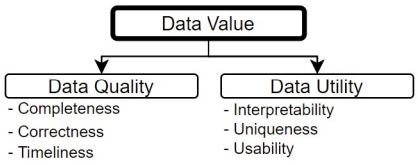


Fig. 3. DMDV taxonomy after dimensions selection

This respondent however recognized that usability may also be included in utility, so that we do not feel the need to update our taxonomy. Our two classes are thus validated.

Our final axis, "identification of potential misclassified dimensions", did not generate any comment as all the respondents agreed with the way the dimensions are classified. Our classification of our dimensions in our classes, and consequently our DMDV taxonomy, are thus validated.

V. FUTURE WORKS

This work, by proposing the concept of Decision-Making Data Value as data selection criterion and building a DMDV taxonomy, is a first necessary step towards a main objective: supporting data selection for decision making. To complete our work and achieve this objective, a lot of future works may be considered. This section discusses the main ones.

A first way to extend our work is to develop a DMDV assessment framework based on our taxonomy. For this purpose, a metric should be developed for each dimension of our taxonomy and a way to aggregate these metrics in a DMDV indicator should be proposed. This indicator of DMDV could be designed at different levels of granularity (e.g. the column level, the database level,...). This would for example allow to rank columns in terms of importance for reporting.

Then, the next step would be to test the proposed framework in real-world situations. More specifically, the choice of the metrics should be tested and adjusted if needed. In order to do this, it would be interesting to develop use cases and to apply the framework to test how it performs in comparison with the assessment of a data specialist. This would also allow to detect some particular cases that are not taken into account.

Finally, a final step would be to integrate our DMDV concept into a data-driven DSS design process. Indeed, this would allow to develop an integrated DSS design process guiding the user in the data selection. This can be quite challenging because it requires to center the data selection part of the DSS design process around the concept of DMDV. This process could then be tested with end-users to discover all the practical possibilities offered by our proposition in a data-driven DSS design process.

VI. Conclusion

In this paper, we tackle the problem of finding a data selection criterion for decision-making support. Organizations have so much data that it becomes nearly impossible for decision makers to intuitively select the most important ones. To address this problem, we suggest to use the decision-making value of data (DMDV) as data selection criterion. We thus define the concept of DMDV and develop a taxonomy of the dimensions having an impact on this concept. For this purpose, we first create a global data value taxonomy from which we derive our DMDV taxonomy, based on four criteria: relevance, distinctiveness, measurability and comprehensiveness. Our taxonomy decompose DMDV dimensions into two classes: data quality and data utility. We present data quality as the combination of completeness, correctness and timeliness dimensions, while data utility is composed of interpretability, uniqueness and usability dimensions. We conduct several interviews to validate our taxonomy and discuss the results. We also elaborate on the future works.

In terms of limitation, even if we try to objectify the selection of dimensions as much as possible, the application of our criteria may include a small part of subjectivity when selecting the final dimension to keep for each subclass. However, these criteria allow to find a set of dimensions that are relevant, distinct, measurable and comprehensive. This means that, even if an other dimension is selected for a particular subclass, its characteristics and meaning should be very similar, resulting in an equivalent set of dimensions.

References

[1] P. Alpar and M. Schulz. Self-service business intelligence. *Business & Information Systems Engineering*, 58(2):151–155, 2016.

[2] D. Arnott and G. Pervan. A critical analysis of decision support systems research revisited: the rise of design science. In *Enacting Research Methods in Information Systems*, pages 43–103. Springer, 2016.

[3] J. Attard and R. Brennan. Challenges in value-driven data governance. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 546–554. Springer, 2018.

[4] J. Attard and R. Brennan. A semantic data value vocabulary supporting data value assessment and measurement integration. 2018.

[5] J. Attard, J. Debattista, and R. Brennan. Saffron: a data value assessment tool for quantifying the value of data assets. 2019.

[6] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[7] M. Bendechache, N. Sudhanshu Limaye, and R. Brennan. Towards an automatic data value analysis method for relational databases. 2020.

[8] C. Bowman and V. Ambrosini. Value creation versus value capture: towards a coherent definition of value in strategy. *British journal of management*, 11(1):1–15, 2000.

[9] R. Brennan, J. Attard, and M. Helfert. Management of data value chains, a value monitoring capability maturity model. 2018.

[10] R. Brennan, J. Attard, P. Petkov, T. Nagle, and M. Helfert. Exploring data value assessment: a survey method and investigation of the perceived relative importance of data value dimensions. In *ICEIS 2019-21st International Conference on Enterprise Information Systems*, pages 200–207. SciTePress, 2019.

[11] L. Cai and Y. Zhu. The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, 2015.

[12] A. Even and G. Shankaranarayanan. Value-driven data quality assessment. In *ICIQ*, 2005.

[13] A. Even and G. Shankaranarayanan. Dual assessment of data quality in customer databases. *Journal of Data and Information Quality (JDIQ)*, 1(3):1–29, 2009.

[14] L. Holst, V. Stich, G. Schuh, and J. Frank. Towards a comparative data value assessment framework for smart product service systems. In *IFIP International Conference on Advances in Production Management Systems*, pages 330–337. Springer, 2020.

[15] M. Kaiser, M. Klier, and B. Heinrich. How to measure data quality?-a metric-based approach. *ICIS 2007 Proceedings*, page 108, 2007.

[16] K. Kannan, R. Ananthanarayanan, and S. Mehta. What is my data worth? from data properties to data value. *arXiv preprint arXiv:1811.04665*, 2018.

[17] C. Lennerholt, J. Van Laere, and E. Söderström. User-related challenges of self-service business intelligence. *Information Systems Management*, 38(4):309–323, 2021.

[18] J. Locke and W. Engels. *Some Considerations of the Consequences of the Lowering of Interest, and Raising the Value of Money*. Verlag Wirtschaft und Finanzen, 2018.

[19] K. Marx. *Le capital*. Librarie du Progres, 1875.

[20] H. S. Neap and T. Celik. Value of a product: A definition. *International Journal of Value-Based Management*, 12(2):181–191, 1999.

[21] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[22] D. J. Power. Decision support systems: a historical overview. In *Handbook on decision support systems 1*, pages 121–140. Springer, 2008.

[23] G. Shankaranarayanan, A. Even, and P. D. Berger. Optimizing data management with disparities in data value. *Journal of International Technology and Information Management*, 24(3):1, 2015.

[24] Statista. Le big bang du big data, 2019.

[25] H. Stein, L. Holst, V. Stich, and W. Maass. From qualitative to quantitative data valuation in manufacturing companies. In *IFIP International Conference on Advances in Production Management Systems*, pages 172–180. Springer, 2021.

[26] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[27] X. Wang, C. Dong, W. Zeng, Z. Xu, and J. Zhang. Survey of data value evaluation methods based on open source scientific and technological information. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 172–185. Springer, 2019.

[28] J. R. Wood and L. E. Wood. Card sorting: current practices and beyond. *Journal of Usability Studies*, 4(1):1–6, 2008.

Data Modeling and Data Analysis in Simulation Credibility Evaluation of Autonomous Underwater Vehicles

Xiaojun Guo, Zhiping Huang, Shaojing Su, Yunxiao Lv

College of Intelligence Science and Technology,
National University of Defense Technology,
Changsha, Hunan, China
jeanakin@163.com; 3421436@qq.com; susj-5@163.com; jeanakin@126.com

Abstract-As an important tool for exploring and defending the ocean, autonomous underwater vehicles (AUVs) play an irreplaceable role. With the help of simulation models, the R&D test cycle of AUV equipment can be accelerated, but the simulation credibility assessment of AUVs faces many challenges: uncertainty, emergence and nonlinearity. This paper starts from the credibility evaluation of the simulation model of AUVs. Based on small-sample judgment criterion, Bayesian Sequential Mess Test (SMT) that makes full use of prior knowledge is proposed for the credibility evaluation of static parameters. For the reliability evaluation of the dynamic simulation model, the NARX steady-state response algorithm and the prior-based identification are used to evaluate the reliability of the dynamic simulation model. The application performance of the data analysis method in the credibility evaluation of AUVs is analyzed.

Keywords-autonomous underwater vehicle(AUV); credibility evaluation; Bayesian sequential mess test; non-linear autoregressive model with exogenous inputs (NARX) steady-state response algorithm

I. INTRODUCTION

As an important tool for exploring and defending the ocean, autonomous underwater vehicles (AUVs) play an irreplaceable role, and simulation credibility evaluation of AUVs has the characteristics of uncertainty, emergence and nonlinearity. Whether the reliability of the simulation model can be effectively evaluated determines the maturity of the AUV technology. AUV systems not only has complex test environmental conditions, but also the complex and randomly changing hydrological conditions under water, the control law based on fluid dynamics, and the limited underwater communication methods have all added certain difficulties to the AUV experiment[1-2]. The number of tests for the evaluation of model parameters and indicators is very limited (tens of times at most). Due to the lack of support from actual flight test data, it is difficult to evaluate the reliability of the simulation model. This problem also constrains a considerable part of AUV M&S research. The reference model used in the verification of the AUV simulation model and the relevant data of the model to be verified are the sink information under a certain channel transmission[3-4]. The uncertainty of the system error and the inaccuracy of the experimental data observation are the main causes of the uncertainty.

An approach for mixture for symmetric distributions was proposed. They focused on the two-component mixture and develop d a Bayesian model using parametric priors for the location parameters and a nonparametric prior based on Dirichlet process [5]. As the similar method handling with small-size sample and high precision, the application of bootstrap is no less than Bayesian method, and also achieved surprising improvement [6]. For the simulation modelling and reliability evaluation of complex large systems, many methods based on expert systems have been proposed[7-8].

In this paper, starting from the reliability evaluation of the simulation model of AUV, firstly, for the reliability evaluation of the static simulation model, a small sample judgment criterion based on precision measurement is given; The Bayesian SMT identification test that makes full use of prior knowledge is used for the reliability evaluation of static parameters; secondly, for the reliability evaluation of the dynamic simulation model, the NARX steady-state response method and the prior-based identification are used to evaluate the reliability of the dynamic simulation model. The structural parameters of the model are identified, thereby transforming the reliability evaluation of the dynamic model into the performance evaluation of the static parameter distribution. Finally, the application performance of the data analysis method in the reliability evaluation of the complex large system is analyzed.

II. DATA ANALYTICS METHODS IN CREDIBILITY EVALUATION OF AUV STATIC SIMULATION MODELS

When analyzing the relevant performance of the small sample test, the traditional statistical method based on classical frequentism has been unable to reasonably explain the test results under the background of the small sample due to its limitations. Most studies use Bayesian statistical methods when the sample size cannot meet the precision requirements of specific applications, but they do not give an exact conceptual definition of small samples [9].

A. Definition of small sample

Based on the application of statistical inference in different contexts, it can be inferred that the size of the sample is judged based on the application. In view of the differences between

DOI reference number: 10.18293/SEKE2022-173

classical frequency statistics and Bayesian statistics, the related concepts of point estimation, interval estimation in classical frequency statistics are used to help explain the definition of sample size.

[Definition 1] A random variable has a density distribution function $f(X)$ .Suppose its variance is σ , the precision required by the application is δ_0 , the point estimation of a certain mathematical characteristic parameter is $\hat{\theta}$, then the precision of this estimation is $\sigma(\hat{\theta}) = \frac{\sigma}{\sqrt{n}}$, n is the sample capacity, then

(1) n satisfies $n > (1/\lambda)ceil(\sigma^2/\delta_0^2), 0 < \lambda < 1$ is the large-sample size of the significance degree $1/\lambda$of the mathematical feature parameter point estimation under the distribution.

(2) n satisfies $n < (1/\eta)ceil(\sigma^2/\delta_0^2), 1 < \eta$ is the small-sample size of the significance degree $\eta$of the mathematical feature parameter point estimation under the distribution.

[Definition 2] A random variable has a density distribution function$f(X)$. When the confidence level is$1-\alpha$, the interval estimation of a certain mathematical characteristic parameter is $[\hat{\theta}(X)-\delta, \hat{\theta}(X)+\delta]$ The required precision is δ_0, $\delta = g(f(X), n)$, n is sample capacity, then

(1) n satisfies $\delta < \lambda\delta_0, 0 < \lambda < 1$ is the large-sample size with a significance degree of $1/\lambda$ in the estimation of the mathematical feature parameter interval under the distribution;

(2) n satisfies$\delta > \eta\delta_0, \eta > 1$ is the small sample size of the significance level η of the mathematical feature parameter interval estimation under the distribution.

B. Bayesian Sequential Mess Test

During World War II, in order to meet the needs of military acceptance work, A. Wald proposed a sequential inspection method, sequential probability ratio test (SPRT). and proved that in all test classes where the probability of making two types of errors does not exceed a given sum of α and β , the average SPRT required test sample (ie, the test sample size) is minimal. Two goals can be achieved with sequential testing as below:

(1) It is expected to reduce the number of tests under the same identification accuracy requirements. The method constructs a buffer region between the rejection region and the acceptance region, avoiding drastically different decisions based on the success or failure of a single trial.

(2) The sampling times can be adjusted according to the current inspection or estimation effect, so that the sample size can be appropriately selected, so that the obtained estimation has a predetermined accuracy; or under a given sampling cost, the risk can be reduced.

Compared with the traditional method, the SPRT method has been greatly improved, and the improvement in reducing the test sample size is significant, but this method does not take into account the prior information, so that the historical test data or empirical data are not fully utilized, and the test sample size is still large. In fact, the model assumptions are usually biased, that is, the robustness of the SPRT method, and the optimality of SPRT is only established under certain hypothetical models.

In the case of fully considering the prior information, based on SPRT, the sequential posterior odd test (SPOT) method is proposed. Given two types of risk upper thresholds (denoted as α_N,β_N), for the truncated test scheme T_N, if these probabilities of the determined decision scheme are within the allowable range, the SPOT truncated scheme $T_N$is judged to be desirable. The solution of the SPOT truncation scheme is transformed into the analysis of the relationship between the decision threshold C and the sample size N and the two types of risks. For the specific application background, the computer-aided method can be used for fitting and solving.

Aiming at the shortcomings of SPRT, the Sequential Mess Test (SMT) method is constructed for the testing scheme of simple hypotheses against simple hypotheses, and it has been proved that it can effectively reduce the test sample size under the condition of equivalent risk. The idea of this method is to split the original two-alternative hypothesis testing problem into multiple groups of hypothesis testing problems under the condition of given two-alternative hypothesis test values p_0, $p_1$and two types of risk upper limit valuesα, β. Taking the SMT hypothesis test with one point inserted as an example, $p_2 \in (p_1, p_0)$ the original SPRT hypothesis test is divided into the following two groups of hypothesis test problems:

$$H_{01}: p = p_2,\ H_{11}: p = p_1$$
$$H_{02}: p = p_0,\ H_{12}: p = p_2$$

For the two sets of hypothesis tests, the SPRT method is used to test them respectively, so that the finite value can be obtained when the algorithm is stopped. Figure 1 depicts an SMT scheme that inserts a point. It can be seen from Figure 1 that the sample size required by this method has an upper bound, when the population distribution tested is a binomial distribution, the upper bound is the intersection of two straight lines.

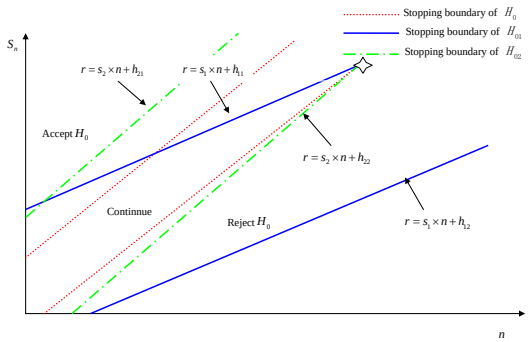


Figure 1. SMT solution for inserting a checkpoint

The minimum sample size of the truncated SMT scheme is also much better than that of the traditional method, but the SMT algorithm that simply inserts multiple points has little improvement in the test effect. This paper combines the above two ideas and constructs a new inspection scheme, which may achieve better improvement effect, which is called the Bayes SMT method. The construction of Bayes SMT test needs to solve the following problems: a) acquisition, quantification and rationality test of prior information; b) Splitting of prior information; c) determination of the principle of Bayes SMT; d) optimal insertion point solution; e) Risk size; f) Truncated program design. As shown in Figure 2, the introduction of SMT test with prior information makes it possible to further reduce the test area, that is, to further reduce the test sample size.

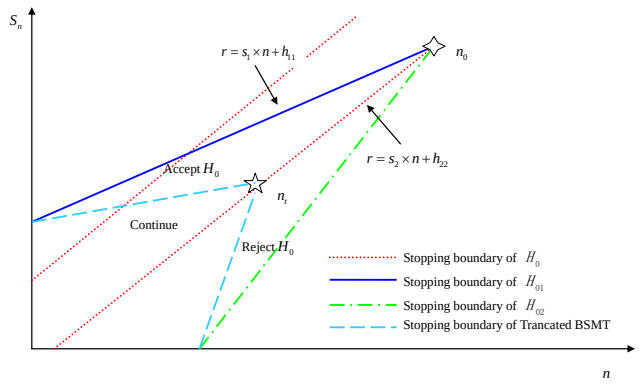


Figure 2. BSMT solution for inserting a checkpoint

Since the starting point of the design of the SMT scheme is the sequential test of simple hypotheses against simple hypotheses, this method is used for multiple hypothesis testing, but cannot well solve the testing problem with complex hypotheses. But in solving the problem of sequential testing of simple hypotheses, it can still better reduce the amount of calculation.

III. Data Modeling and Data Analysis Methods in the Credibility Evaluation of Dynamic Simulation Models

A. *Steady-state response method for NARX model*

In the early stage of nonlinear system identification theory, a general nonlinear regression model with exogenous variables (non-linear autoregressive model with exogenous inputs, NARX) was proposed. As a universal model of NARX, NARMAX (non-linear autoregressive moving average models with exogenous inputs) model basically covers almost all nonlinear models such as bilinear models, H-models, W-models, nonlinear time series models, ARMAX models, etc. As illustrated in Figure 3.

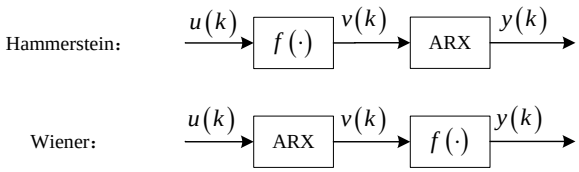


Figure 3. Block-connected nonlinear characterization of Hammerstein and Wiener models

The general NARX model can be expressed as

$$y(k) = F^{\ell}\left[y(k-1), \cdots, y(k-n_y), u(k-d), \cdots, u(k-n_u), e(k)\right] \quad (1)$$

Equation (1) can be expanded into a polynomial sum of nonlinearity in the interval$[1, \ell]$,The $(p+m)$th term includes a p-order $y(k-n_i)$, an m-order $u(k-n_i)$, and a multiple factor$c_{p,m}(n_1, \cdots, n_m)$, as shown in Equation (2).

$$y(k) = \sum_{m=0}^{\ell} \sum_{p=0}^{\ell-m} \sum_{n_1,n_m}^{n_y,n_u} c_{p,m}(n_1, \cdots, n_m) \prod_{i=1}^{p} y(k-n_i) \prod_{i=1}^{m} u(k-n_i) + e(k) \quad (2)$$

B. *NARX characterization of AUV motion models*

The nonlinear state equation of the motion model of the 6-DOF AUV can be expressed as

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} = g(\boldsymbol{x}) \end{cases} \quad (3)$$

Where,

$\boldsymbol{x} = \left[v_x, v_y, v_z, w_x, w_y, w_z, x_b, y_b, z_b, \psi, \theta, \phi\right]^T$, $\boldsymbol{u} = f\left(v_x, v_y, v_z, w_x, w_y, w_z, \delta_E, \delta_R, \delta_D\right)$,

$$e_1 = m\left(v_y w_z - v_z w_y\right) + mx_G\left(w_y^2 + w_z^2\right) - my_G w_x w_y - mz_G w_x w_z + \lambda_{22} - \lambda_{33} v_z w_y + \lambda_{26}\left(w_y^2 + w_z^2\right) - \left(mg - F_G\right)\sin\theta + Te + Rx$$

$$e_2 = m\left(v_z w_x - v_x w_z\right) + my_G\left(w_x^2 + w_z^2\right) - mz_G w_y w_z - mx_G w_x w_y + \lambda_{33} v_z w_x - \lambda_{11} v_x w_z - \lambda_{26} w_x w_y - \left(mg - F_G\right)\cos\theta\cos\varphi + Ry$$

$$e_3 = m\left(v_x w_y - v_y w_x\right) + mz_G\left(w_x^2 + w_y^2\right) - mx_G w_x w_z - my_G w_y w_z + \lambda_{11} v_x w_y - \lambda_{22} v_y w_x - \lambda_{26} w_x w_z + \left(mg - F_G\right)\cos\theta\sin\varphi + Rz$$

$$e_4 = \left(I_{yy} + \lambda_{55} - I_{zz} - \lambda_{66}\right) w_y w_z + my_G z_G\left(w_y^2 - w_z^2\right) + mx_G z_G w_x w_y - mx_G y_G w_x w_z + my_G\left(v_x w_y - v_y w_x\right) + mz_G\left(v_x w_z - v_z w_x\right) + mgy_G\cos\theta + mgz_G\cos\theta\cos\varphi + Mx$$

$$e_5 = \left(I_{zz} + \lambda_{66} - I_{xx} - \lambda_{44}\right) w_x w_z + mx_G z_G\left(w_z^2 - w_x^2\right) + mx_G y_G w_y w_z - my_G z_G w_x w_y - mgz_G\sin\theta + my_G\left(v_y w_z - v_z w_y\right) + mx_G\left(v_y w_x - v_x w_y\right) + \lambda_{26}\left(v_y w_x - v_x w_y\right) + mgx_G\cos\theta\sin\varphi + My$$

$$e_6 = \left(I_{xx} + \lambda_{44} - I_{yy} - \lambda_{55}\right) w_x w_y + mx_G y_G\left(w_x^2 - w_y^2\right) + my_G z_G w_x w_z - mx_G z_G w_y w_z + mgy_G\sin\theta + mx_G\left(v_z w_x - v_x w_z\right) + my_G\left(v_z w_y - v_y w_z\right) + \lambda_{26}\left(v_z w_x - v_x w_z\right) - mgx_G\cos\theta\cos\varphi + Mz$$

$\boldsymbol{E} = [e_1, e_2, e_3, e_4, e_5, e_6]^T$, $\boldsymbol{M} = \boldsymbol{M}_I + \boldsymbol{M}_A$,

$\left[\dot{v}_x, \dot{v}_y, \dot{v}_z, \dot{w}_x, \dot{w}_y, \dot{w}_z\right]^T = \boldsymbol{M}^{-1}\boldsymbol{E}$,

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix} = \boldsymbol{C}_0^b \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix},$$

$$\begin{bmatrix}\dot{\psi}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix}\left(w_y\cos\phi-w_z\sin\phi\right)/\cos\theta\\ w_y\sin\phi+w_z\cos\phi\\ w_x-\left(w_y\cos\phi-w_z\sin\phi\right)\tan\theta\end{bmatrix}$$

TABLE I. PARAMETERS AND DEFINITIONS IN AUV MOTION MODEL

Variable	Definitions
m	AUV quality (kg)
S	Characteristic area, generally take the largest cross-sectional area (m^2)
L	Feature length, generally take the total length of AUV (m)
v	AUV speed(m/s)
$\delta_E,\delta_R,\delta_D$	Horizontal rudder, straight rudder, differential rudder
ρ	The density of water, here $994kg/m^3$
$c_x(\cdot),c_y(\cdot),c_z(\cdot)$	Drag, lift, side force coefficients
$m_x(\cdot),m_y(\cdot),m_z(\cdot)$	Roll moment, yaw moment, pitch moment coefficient
x_G,y_G,z_G	Center of gravity backward shift, descent, side shift
I_{xx},I_{yy},I_{zz}	Inertia torque in X, Y, Z directions
λ_{ij}	Component coefficients for additional inertial forces and moments
α,β	Angle of attack, angle of sideslip
v_x,v_y,v_z	Velocity components in X, Y, Z directions
w_x,w_y,w_z	Angular velocity components in the X, Y, Z directions
x_b,y_b,z_b	The position component of the AUV in the ground coordinate system
ψ,θ,ϕ	Yaw angle, pitch angle, roll angle
D_t	AUV displacement
g	Gravitational acceleration, here take $9.81(m/s^2)$

So far, the mechanism modeling of the functional relationship$\dot{\boldsymbol{x}}=f(\boldsymbol{x},\boldsymbol{u})$ has been completed. Taking the observation equation$\boldsymbol{y}=g(\boldsymbol{x})=\dot{\boldsymbol{x}}$, there is a discretized NARMAX state equation as

$$\boldsymbol{X}(k+1)=F^{\ell}(\boldsymbol{X}(k),\boldsymbol{U}(k),\boldsymbol{C}_0^b,Te,Rx,\cdots,Mx,\cdots) \tag{1.1}$$

For Equation (1), the function $f(\cdot)$between the steady-state intermediate signal $v(k)$ and the output signal $y(k)$ is steady-state, and it is the gain of the ARX model. If the steady-state gain is adjusted to 1, that is$\bar{v}(k)=\bar{u}(k)$, it is the difference between the input and the output, the steady-state response function$f(\cdot)$. The function can be used to obtain the corresponding relationship of $\bar{y}\times\bar{u}=\bar{v}$, and a certain linear regression method can be used to obtain the function estimate$\bar{v}=g(\bar{y})$, which is an estimate on the application domain. Assuming that the model (shown in equation (2)) is excited by a constant input, then the steady-state response is

$$\bar{y}=y(k-1)=y(k-2)=\cdots=y\left(k-n_y\right),\bar{u}=u(k-1)=u(k-2)=\cdots=u(k-n_u) \tag{4}$$

Then Equation (2) can be rewritten as

$$y(k)=\sum_{m=0}^{\ell}\sum_{p=0}^{\ell-m}\sum_{n_1,n_m}^{n_y,n_u}c_{p,m}(n_1,\cdots,n_m)\,\bar{y}^p\bar{u}^m \tag{5}$$

This gives the model response at a specific input point.

C. NARX Model Steady-State Response Method for Grey Box Identification

The method of parameter identification using the steady-state response of the SISO-NARX model can be extended to multi-dimensional situations, and the same example is still used for analysis. This nonlinear system is the MISO-NARX system. The steady-state response parameter identification method of NARX is based on The SISO system proposes that the labels of variables in these methods are all defined based on SISO. When the identification of MISO-NARX is realized, the relevant labeling methods need to be improved. It can be known from the combination function$P_i(k)$,

$$y_0(k)=\sum_{i=1}^{12}d_i f^{\ell}\left(v_x,v_y,v_z,w_x,w_y,w_z,\theta,\phi,\dot{v}_y,\dot{w}_x,\dot{w}_z,R_y\right)+d_0 \tag{6}$$

In Equation (6), there are a total of 12 inputs, although there is no input in the form of $y_0=\dot{v}_x$itself, but is a component of the multi-dimensional output

$\boldsymbol{Y}=\left[\dot{v}_x,\dot{v}_y,\dot{v}_z,\dot{w}_x,\dot{w}_y,\dot{w}_z,\dot{x}_b,\dot{y}_b,\dot{z}_b,\dot{\psi},\dot{\theta},\dot{\phi}\right]^T$,

$n_y=n_u=1$,so the nonlinear NARX relationship with the above 12 inputs needs to be considered when

$u_i(k-1)=u_i(k)$.

From the test data without identification design, select the data that meets the conditions for steady-state response identification, that is, the steady-state response process. The 100 data samples used in the above two algorithms come from a direct flight and speed-up process. Therefore, except when $v_x(k-1)=v_x(k)=15.2408$ entering a steady state, all other observed variables are 0 or close to 0. At this time, there are

$$-\varepsilon<(mg-\rho D_t g)\left(1+\frac{\theta^2}{2}\right)\left(1+\frac{\phi^2}{2}\right)-\bar{R}_y<\varepsilon \tag{7}$$

The steady-state input referred to$\bar{R}_y$ in Equation (7) , ε is the identification accuracy requirement, and the steady-state output of at this time is -310.82, which is why the identification parameter $mg-\rho D_t g$ always has a small identification variance no matter which identification algorithm is used. Figure 4.5 shows part of the time series of test data that can be used for identification. For the convenience of observation, the data has been transformed such as translation and compression. The identification carried out by Equation (7) is carried out by intercepting a meta-process or meta-process segment in the test. Searching for the segment that can be used for steady-state response identification in the whole process, the identification segment and identification inequality can be obtained, as shown in Figure 4.

As shown in Table II, the identification of the first two methods uses the sampling data of 0-40s, and the sampling period is 0.025s. It can be seen from the table that for this example, the effect of parameter identification is NARX steady-state corresponding method RFF-LS RPEA-BP. The forgetting factor algorithm of RFF-LS gives priority to new information to prevent parameter identification drift caused by data saturation, and makes full use of prior information and experimental collection of new information. The BP neural network recognizes the internal structure and parameters of the sample system through training, but for the model system whose structure is known, the accuracy of the parameter identification results is poor. The grey-box identification method based on the NARX steady-state response makes full use of the characteristics of the input-output relationship of the system's steady-state response in the NARX model. Although there is no special experimental design requirement for the belt identification system, a sufficient amount of steady-state response identification inequality is indispensable.

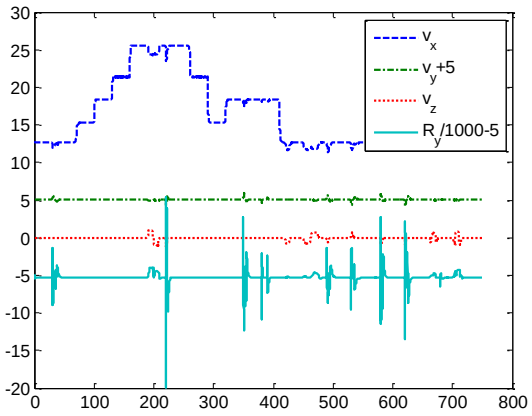


Figure 4. Partial identification data

TABLE II. THREE ALGORITHMS BASED ON THE SAME PRIOR CONSTRAINTS TO IDENTIFY THE FITTING ACCURACY OF PARAMETERS

Period	Accumulation of identification errors		
	RFF-LS	RPEA-BP	Steady state response method
0-40s	6.5254	14.5911	5.5376
0-200s	132.851	213.284	78.265
0-400s	416.4397	520.515	222.549
0-750s	814.675	1302.32	416.61

The parameters identified in different field tests are $\theta_i(i = 1,2,\cdots,N)$, where N is the number of field tests that can be used for grey box identification, and the model parameter based on mechanism modeling is θ_0, at this time, the reliability of the dynamic model based on grey box identification is transformed into a test of whether it follows the distribution, or whether it falls within the estimated mean interval with a certain confidence level. For θ_i , assuming that it obeys a normal distribution, estimate the Bootstrap BCa interval estimate under each confidence level. Then examine the placement points, and measure the credibility level of the virtual model (mechanism modeling) with confidence.

IV. CONCLUSION

Based on the test accuracy requirements, the criterion of sub-sample capacity traits is given, which provides a quantitative standard for the determination of sub-sample size; when conducting the sequential hypothesis testing of small sub-sample test index parameters, reference is made on the basis of SPRT, SPOT, and SMT. Bayes theory proposes the Bayes SMT test method, which can theoretically save the sample size based on the prior; a grey box parameter identification method for the NARX model based on prior and steady-state response is proposed. The identification results show that this method improves the parameter identification of the AUV equation of motion accuracy and identification efficiency. On the basis of grey box identification, using prior distribution information and modern statistical inference methods, the reliability assessment of dynamic models is transformed into the reliability assessment of static parameter models, which provides a method for credibility assessment of dynamic models with prior information. new way.

REFERENCES

[1] Hyakudome, T., Design of Autonomous Underwater Vehicle[J]. International Journal of Advanced Robotic Systems, 2011. 8(1): 122-130.

[2] Zhou J. and Wang S.,Dynamics Modeling and Maneuverability Simulation of the Unmanned Underwater Vehicle Hanging Torpedoes Externally[C]. International Asia Conference on Informatics in Control, Automation and Robotics. 2009.

[3] Yanhui Wei, Yongkang Hou, Shanshan Luo.Combined dynamics and kinematics networked fuzzy task priority motion planning for underwater vehicle-manipulator systems[J]. International Journal of Advanced Robotic Systems 2021 Vol.18 No.3:1729-1814

[4] Guo X.J., et al., Validation of Torpedo Control System Simulation Models on Grey Relational Analysis[C]. ICIECS 2010.

[5] Pituch KA, Joshi M, Cain ME. The Performance of Multivariate Methods for Two-Group Comparisons with Small Samples and Incomplete Data[J]. Multivariate behavioral research 2020 Vol.55 No.5 P704-721

[6] Koller, M. and W.A. Stahel, Sharpening Wald-type inference in robust regression for small samples[J]. Computational Statistics and Data Analysis, 2011. 55: 2504-2515.

[7] Min, F.Y., et al., Knowledge-based method for the validation of complex simulation models[J]. Simulation Modelling Practice and Theory, 2010. 18: 500-515.

[8] Sinisi, Stefano , Alimguzhin, Vadim , Tronci, Enrico . Reconciling interoperability with efficient Verification and Validation within open source simulation environments[J].Simulation Modelling Practice & Theory 2021 Vol.109 1569-190X

[9] Bové D.S. and Held L., Bayesian fractional polynomials[J]. Stat Comput, 2011. 21: 309-324.
[10] J. Raimbault, D. Pumain,Exploration methods for simulation models[J].Physics 2019:5:1-16
[11] Li, Fang, Goerlandt, Floris. Numerical simulation of ship performance in level ice: A framework and a model[J].Applied Ocean Research 2020 Vol.102,1181-1187
[12] Liu, Chao a Zhanhong b, Sarkar, Soumik H. b. Root-cause analysis for time-series anomalies via spatiotemporal graphical modeling in distributed complex systems[J].Knowledge-Based Systems 2021 Vol.211,1057-1063

Improving Database Learning with an Automatic Judge

Enrique Martin-Martin*, Manuel Montenegro†, Adrián Riesco‡, Rubén Rubio§
Dpto. Sistemas Informáticos y Computación, *Universidad Complutense de Madrid*, Madrid, Spain
Email: *emartinm@ucm.es, †montenegro@fdi.ucm.es, ‡ariesco@ucm.es, §rubenrub@ucm.es

***Abstract*—Databases are a key subject in several technical degrees. Because they have a strong practical nature, students require a large number of problems to master them. However, these problems are useful only if accurate and timely feedback is provided. In this paper, we present the learning improvements obtained by using LearnSQL, an automatic judge that has been designed to complement face-to-face lectures. We have measured the impact of this judge during the 2021/22 academic year and report promising results both in student engagement and final grades.**

***Index Terms*—Database Learning, Automatic Judge, SQL**

I. Introduction

Databases and data-processing systems are key topics in many technical degrees (see, e.g., the computer science curricula [1]). In fact, databases are ubiquitous and being able to manage them is a basic ability required by all companies. Database lectures include both theoretical aspects, in particular database design, and practical aspects, which include creating, searching, and indexing data. We will focus on these practical aspects, which score higher in Bloom's taxonomy [2]. In this sense, it is very important to provide the students with (i) as many programming exercises as possible and (ii) immediate feedback, so they are not blocked when an error occurs. However, although it is possible to generate a large amount of exercises, the high number of students in standard classrooms prevents instructors from giving this feedback timely. For this reason, it is important to provide automatic means to assess exercises, but it is also important that this feedback is as informative as possible, because the users are students that have not mastered the subject yet.

Automatic assessment [3] is a well-known field that tries to solve this problem by automatically generating a reply (the type of reply depends on the technique) to students' exercises. Among these techniques, we are interested in automatic judges [4], [5], which were initially developed for quickly evaluating programming exercises in competitions [6]. These judges, which only produced a plain correct/wrong answer, have been adapted to different environments (mainly to teaching of programming languages), easing the evaluation of a large number of students in a short time. In our case, we are not concerned with evaluation but with teaching, so we must provide a collection of exercises, including timely and concrete feedback, as large as possible. In this sense, we will take the main ideas of these systems and enrich them to focus on learning.

DOI reference number: 10.18293/SEKE2022-025

In this paper, we present the learning improvements obtained by using LearnSQL in our database course. LearnSQL is an automatic judge designed to complement face-to-face lectures with a wide set of SQL problems that students can try at any time. The judge supports not only problems about SQL queries but also procedural SQL problems where the student is expected to define procedures, functions, and triggers, so it covers a large part of the course syllabus. Moreover, LearnSQL provides detailed feedback pointing to the source of the error, so students can understand their mistakes and fix their code. When measuring the learning improvements, we are mainly interested in finding evidence that students that use the judge obtain better grades than those who do not. However, we are also interested in discovering the usage degree and student engagement with the judge.

The rest of the paper is organized as follows: Section II presents the context of the subject where LearnSQL is used. Section III introduces the judge, while Section IV evaluates the effects on the students' performance. Finally, Section V concludes and presents some topics of future work. The source code of LearnSQL is available at https://github.com/emartinm/lsql.

II. Context of the Course

We have used LearnSQL in the introductory course on databases which is part of the degree programs at the Faculty of Computer Science of the Complutense University of Madrid, Spain. These degree programs cover 4 years, being the databases course taught in the first semester of the 2nd year. Specifically, for this evaluation of the learning improvement, we used the judge in the academic year 2021–2022, i.e., from September 2021 to January 2022. The syllabus of the introductory course on databases covers the standard contents [7], [8]: relational model, entity-relationship model, SQL queries, procedural SQL (functions, procedures, and triggers), and transactions. The teaching in the introductory course on databases is face-to-face, organized in 30 lectures of 100 minutes each. Approximately 50% of the lectures are theoretical and the other 50% are practical sessions where the students solve exercises that require performing SQL queries and defining functions, procedures, or triggers in the database. In the year 2021–2022 there were 6 different groups in this course, with 40–80 students in each group.

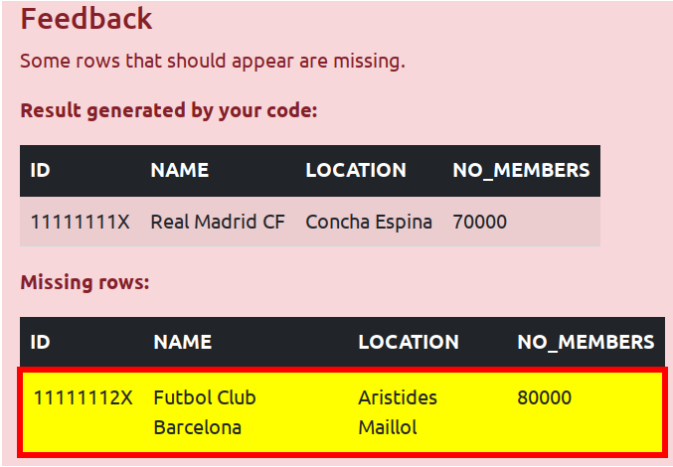


Fig. 1. LearnSQL feedback marking missing rows

For the evaluation of LearnSQL, we have used the judge in 3 groups of the course, taught by two different teachers, totaling 157 students. All the students in these selected groups have had access to the judge and all the exercises from the beginning of the course, concretely 166 problems about SQL queries and 23 problems about procedural SQL (functions, procedures, and triggers). We introduced the judge to the students in the first weeks of the course, and we encouraged them to use it to solve the exercises of the practical classes. However, the use of LearnSQL was completely voluntary for the students: the assignments in the practical classes were free practice exercises that were not assessed and did not have any impact on the final grade of the course. In other words, all students could freely use LearnSQL as a tool to practice SQL if they considered it was useful for them.

The 70% of the subject grade was obtained on the basis of a final written exam. This exam, with a total mark of 10 points, was composed of 3 parts. The first part covered the design of databases for a total of 3.5 points. The second part was the main component of the exam, with exercises about SQL queries and procedural SQL up to a total of 6 points. This is the part of the exam we will focus on when evaluating the impact of LearnSQL on students' learning (see Section IV). Finally, there was a final part about transactions with a value of 0.5 points.

III. LearnSQL

LearnSQL is an automatic judge for database problems. It is open-source software available at https://github.com/emartinm/lsql/ under the MIT license, so any instructor can deploy it in their server and adapt it to their needs. LearnSQL provides a simple web interface where students can browse the collections and problems, as well as submit their solutions by writing code in a text area. One of the key design features of LearnSQL is that it is an automatic judge *for learning*, so it provides detailed feedback when the student submits an incorrect solution. This detailed information helps students to understand their mistakes and fix their solutions, reinforcing positively their learning process. The received feedback ranges from syntax errors to differences in the schema of the solution (different number of columns, or wrong datatypes or names) or a detailed list of missing or incorrect rows, as shown in Figure 1.

LearnSQL supports several types of problems that are suitable for an introductory databases course:

- SQL queries: the student is asked to provide an SQL query that returns some expected results from the database.
- DML sentences: the student's code is expected to produce changes in the database by adding, removing, or updating rows.
- Function and procedure definitions: the judge asks for the definition of a function or procedure fulfilling some expected behavior when invoked.
- Triggers: the student has to write a trigger linked to a table manipulation, which must perform some modifications in the database.
- Discriminate SQL queries: the judge shows two SQL queries to the student that are very similar but not equivalent. The goal is that the student has to provide a set of rows in the involved tables such that the queries produce different results. This kind of problems is very interesting because it requires students to reach the highest levels of Bloom's taxonomy [2], as they must evaluate the queries and create a database state that could differentiate them.

To verify the correctness of a student's submission LearnSQL follows a simple approach: it executes the student's code in the DBMS and compares the obtained results to the expected ones generated by the official instructor's solution. This comparison can be done with more than one test case to increase the accuracy of the judgment. Currently, LearnSQL only supports Oracle 11g as DBMS for executing SQL code because is the system used in our introductory databases course. However, the SQL execution component follows a clear interface and could be easily replaced to connect to any other relational DB system like PostgreSQL or MySQL.

Apart from the judgment based on execution, LearnSQL also provides a richer feedback obtained by a semantic analysis of the student code. For that, it relies on the Datalog Educational System [9] (DES). DES is a deductive database system that supports many formats and query methods: Datalog, Relational Algebra, Tuple Relational Calculus, Domain Relational Calculus, and SQL. When handling a query, DES performs a set of complex analyses on the code that can detect several semantic errors [10], e.g. unnecessary joins, inconsistent or tautological conditions, or unnecessary subqueries. This information is very relevant to students as it allows them to not only fix but also improve and simplify their SQL code, and gain more knowledge about their solution.

Finally, LearnSQL has been designed to be multilingual, so it could be used in any teaching environment. At the moment it only supports Spanish and English, but it could be easily adapted to any other language by providing a text file with the translation of all the strings used in the system. From the point

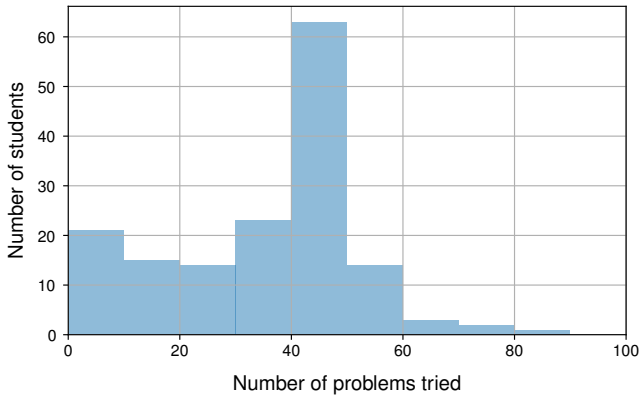


Fig. 2. Problems tried by student

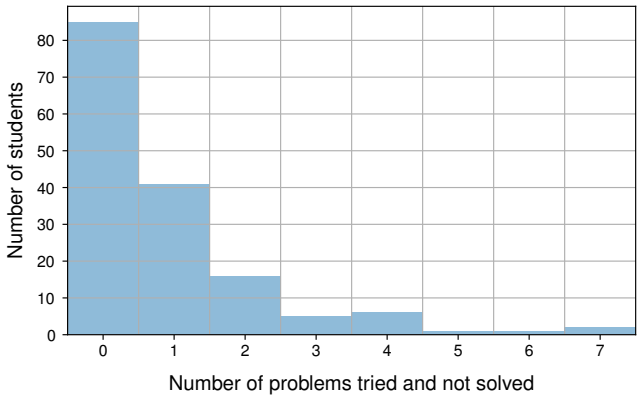


Fig. 3. Problems tried but not solved by student

of view of motivation and linked to gamification [11], LearnSQL supports an interesting feature: achievements. Teachers can define achievements that students obtain when solving a certain number of problems or collections, or a given number of problems of a concrete type. Apart from the satisfaction of obtaining recognition for their work, the achievements are shown as badges in the global ranking, fostering a healthy competition in the classroom.

IV. Evaluation

In this section, we first discuss how the judge was used according to its activity logs. Combining this information with the marks of the final assessed exam, we analyze whether there is quantitative evidence to support that the judge has had a positive effect on the students' performance and learning process.

Out of the 157 students registered in the course described in Section II, we have limited our analysis to the 130 who have attended the final exam, since their marks are useful for our analysis. However, among those who have not attended this exam, 15 have also practiced with LearnSQL, solving 25.8 problems in mean. The students have tried an unequal amount of exercises among the 189 available in the judge, as shown in Figure 2, and effectively solved the majority of those tried. The latter is illustrated in Figure 3, where it can be seen that just a few problems were left unsolved (7 at most, but most of the students left between 0 and 2), with an average of 8.18 attempts before giving up. The number of attempts for a student to solve a given problem is usually low ($\mu = 2.57$, $\sigma = 3.88$), the first try has been enough in the 60.54% of the cases, and the second attempt in a 15.64%. However, except for a student with a single submission, everyone has failed some attempts and obtained feedback from the judge.

We realize that most students have tried to solve around 40 exercises ($\mu = 38.78$, $\sigma = 16.28$), and only 8 of them have not used LearnSQL at all. Around 30 students have used the judge throughout the whole course duration, but the majority have concentrated their interaction during the first to second month of the lessons, two months before the exam. This time window coincides with the time when SQL queries and procedural SQL

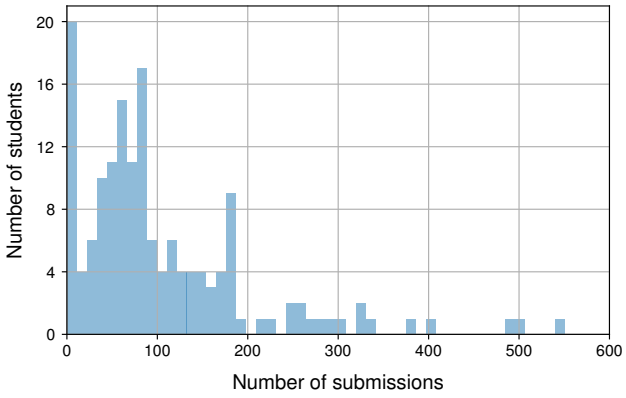


Fig. 4. Problems submitted by student

are taught in class. However, shortly before the final exam, the number of submissions has slightly increased. 85.91% of the submissions have been done outside class hours. It is also interesting to consider the information in Figure 4, which indicates that the judge has been used extensively. Although some students have not used the judge at all, those that have used it have submitted many times, and some of them sent more than 500 submissions. Combining this information with the one presented in Figure 3, we consider that students are engaged by the judge and do not stop until the problems have been solved.

In order to evaluate the effect on learning of LearnSQL, we compare the usage profile of the students with the marks they have obtained in the SQL-related exercises of the final exam. Figure 5 shows the distribution of those marks aggregated into five equally numerous subsets by increasing number of tried problems, where we can observe an increasing trend on the marks. The Spearman's and Pearson's correlation coefficients between these metrics are respectively 0.497 ($p = 1.8 \cdot 10^{-9}$) and 0.457 ($p = 4.6 \cdot 10^{-8}$), so there is statistically significant (linear) relationship between them. In summary, we consider that practicing with LearnSQL has a positive effect that becomes more noticeable as the amount of practice increases, but it is also observable with relatively little training.

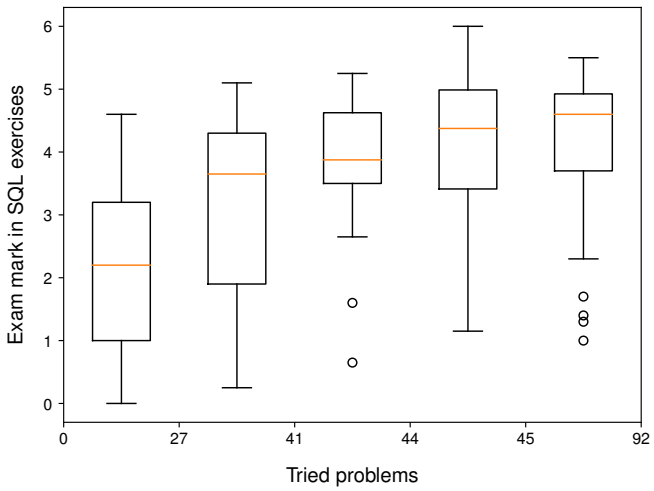


Fig. 5. Mean mark by tried problems

Regarding specific categories of exercises, the students who solved at least one problem on procedures in the judge have obtained a mean mark of 1.59 out of 2, while the mean mark of the rest is 0.99. Similarly, those who solved the five problems on triggers have obtained a mark of 0.95 out of 1 on average, compared to 0.45 for those who have not tried any problem of this kind.

We should acknowledge some threats to the validity of these conclusions. Since the usage of the judge was discretionary for the students, its hypothetical effect on the performance in the final exam could alternatively be explained by the general intuition that stronger and more enthusiastic students participate more in the activities of the course and also obtain better marks. However, the Spearman's correlation between the number of problems solved and the marks of exam exercises not related to SQL is much lower ($\rho = 0.25$, $p = 0.003$). Even if we assume that students have gained SQL skills by practicing with LearnSQL, we may wonder whether this same improvement could have been obtained with the classical non-assessed exercises, but more complex experiments would be required to analyze this.

V. Conclusion and Ongoing Work

In this paper, we have presented the learning improvements obtained by using LearnSQL in our database teaching. LearnSQL is a judge for automatically assessing the correctness of database exercises, including SQL queries, triggers, and procedures, among others. Unlike other automatic judges, LearnSQL has been designed to provide detailed explanations to students in order to help them detect their errors. LearnSQL has been used during the 2021/22 academic year and its usefulness has been measured. The results are promising, as there is statistical evidence that using the judge improves the final degree obtained by the students.

As future work, we plan to integrate the judge into Moodle, the learning platform used in our virtual campus, following an approach similar to the one in [12]. In this way, it would use LearnSQL to (possibly partially) validate assignments from the students and to integrate the marks obtained using our judge with the rest of marks, as well as presenting the students a single source for the whole subject.

In order to obtain further insight into the learning benefits of LearnSQL, we consider repeating the evaluation next year including other groups where classical non-assessed exercises are proposed to the students as control groups. Moreover, for confirming the conjectured positive effect of the informative feedback provided by the judge, we plan to offer to a separate student group a restricted version of LearnSQL where only a binary answer (correct/wrong) is obtained.

Finally, it might be interesting to add more problems and to test the tools in different degrees. In particular, we are interested in analyzing the impact of using this kind of judge with the students from the Mathematics degree.

Acknowledgements

This work has been supported by the teaching innovation projects of the Complutense University of Madrid INNOVA-Docencia 2020-21/18 and 2021-22/387.

References

[1] "Computing Curricula 2020," December 2020, https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf.

[2] B. S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. David McKay Company, 1956.

[3] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of recent systems for automatic assessment of programming assignments," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling 2010. ACM, 2010, pp. 86–93.

[4] N. Gerritsen, T. Kinkhorst, and T. Werth, *DOMJudge 8.0 manual*, DOMJudge, 2022.

[5] L. Llana, E. Martin-Martin, C. Pareja-Flores, and J. Á. Velázquez-Iturbide, "FLOP: A user-friendly system for automated program assessment," *J. Univers. Comput. Sci.*, vol. 20, no. 9, pp. 1304–1326, 2014. [Online]. Available: https://doi.org/10.3217/jucs-020-09-1304

[6] M. A. Revilla, S. Manzoor, and R. Liu, "Competitive learning in informatics: The UVa online judge experience," *Olympiads in Informatics*, vol. 2, pp. 131–148, 2008.

[7] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems (7th Edition)*. Pearson Harlow, 2017.

[8] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database system concepts (7th edition)*. McGraw-Hill, 2019.

[9] F. Sáenz-Pérez, "DES: A deductive database system," *Electron. Notes Theor. Comput. Sci.*, vol. 271, pp. 63–78, March 2011. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2011.02.011

[10] F. Sáenz-Pérez, "Applying constraint logic programming to SQL semantic analysis," *Theory and Practice of Logic Programming*, vol. 19, no. 5-6, p. 808–825, 2019. [Online]. Available: http://dx.doi.org/10.1017/S1471068419000206

[11] J. Hamari, *Gamification*. John Wiley & Sons, Ltd, 2019, pp. 1–3. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405165518.wbeos1321

[12] J. L. Brita-Paja, C. Gregorio, L. Llana, C. Pareja, and A. Riesco, "Introducing MOOC-like methodologies in a face-to-face undergraduate course: a detailed case study," *Interactive Learning Environments*, vol. 27, no. 1, pp. 15–32, 2019. [Online]. Available: https://doi.org/10.1080/10494820.2018.1451345

Data Regulation Ontology

Guillaume Delorme, Guilaine Talens, Eric Disson
Laboratoire Magellan
University Jean Moulin Lyon 3, Iaelyon School of Management
Lyon, France
@univ-lyon3.fr

Guillaume Delorme
Group Security
Solvay
Lyon, France
@solvay.com

***Abstract*—*The recent upsurge enactment of regulations seeking to regulate data processing induces a complexification of compliance management for regulated firms. Firms wishing to implement efficient, cost effective and compliant information security and risk management require an increased comprehension of regulatory requirements. Following a previous paper defining Data Regulation Risk, this paper describes an ontology to apprehend the business and operational impacts of regulatory requirements. The ontology is structured to handle various firms' legal context while remaining agnostic of risk management methodologies.*

Keywords-Ontology; Compliance; Knowledge-based; Privacy

I. INTRODUCTION

Over the past decades, the upsurge enactment of regulations seeking to reinforce the protection of individuals' rights and privacy, economic interests and national security has led to the appearance of a new class of risk called Data Regulation Risk (DRR) [1]. We defined Data Regulation as a norm governing data processing and/or ICT governances and processes and/or information technologies and services. Despite addressing similar concepts, such regulations are yet often demanding divergent or particular controls. This article aims to furnish the necessary information to facilitate DR management.

Several authors pointed out the need for ontology in the security domain [2, 3]. Similarly, several conceptualizations of the legal domain have been presented and studies and comparison of legal ontologies can also be found [4, 5]. Despite important contributions, there is a need for methodologies and models to identify multi-disciplinary risks like DR management. We seek to address DR by building an ontology which facilitates its management.

Ontologies are designed to facilitate the sharing, use and re-use of knowledge [6]. Defined as explicit conceptualization of a domain [7], they enable its modulization with the desired level of abstraction depending on the initial objective. We develop an ontology following the Enterprise Model Approach [8] with the ambition of facilitating the apprehension of business and operational impacts of regulatory requirements. It focuses on regulatory controls while leaving the option of mapping the controls with additional threats for a broader or multi-disciplinary risk management. To reach our target, we use to the extent possible the terminologies of the WordNet database developed by Princeton University [9] as well as concepts present in existing ontologies.

This contribution is structured as follows. Section 2 discusses the core ontology and its purpose. Section 3 presents the building and usability of the ontology. Finally, section 4 draws conclusions and discusses some further research directions.

II. THE CORE ONTOLOGY ARCHITECTURE AND ITS KEY CONCEPTS

The creation of an ontology requires to determine what entities should be considered and studied.

A. Methodology

With the ambition of easing methodology building, [6] surveyed different ontology building methodologies such as TOVE [10], Enterprise Model Approach [8], Methontology [11] and Ontolingua [7]. Recent methodologies have been developed to focus on specific needs such as [12]. As no methodology seems to stand out and all of them have their pros and cons [5], we decided to adopt the Enterprise Model Approach which is a stage-based approach, widely spread, providing sufficient freedom of representation [13]. It is appropriate to a cross disciplinary ontology such as ours and is articulated around four main stages : identify purpose, building the ontology, evaluation and documentation. The second stage incorporates the ontology capture, ontology coding and the integration of existing ontologies [8].

As opposed to the classic bottom-up and top-down approach to identify the main terms of our ontology, we opt for the middle out approach presented in [8]. This approach allows one to identify the primary concepts of the ontology before moving on to specialize or generalize terms [11]. The middle out approach implicitly leads to more stable concepts. In regards to clarity, which is the foundation of the usability and reusability of an ontology, we need a world known, easily accessible, proven and accepted terminology database. Suggested Upper Merged Ontology (SUMO) [14] is a formal public ontology providing definitions for general purpose terms and is intended as a unifying framework for more specific domain level ontologies. As SUMO is designed as an upper ontology, it provides generic terms and therefore fails to address the needs of more specific domains ontologies [15]. We then decide to use when possible the terminologies of the WordNet database developed by Princeton University [9]. WordNet "is a large lexical database of nouns, verbs, adjectives and adverbs grouped into sets of cognitive synonyms (Synsets), each expressing a distinct concept. Synsets are interlinked by

DOI reference number: 10.18293/SEKE2022-024

means of conceptual-semantic and lexical relations." Each concept, relation or attribute in our ontology is mapped with a unique Synset using the Synset ID.

B. Purpose

Defining an ontology purpose and what its intended uses are, is the fundamental step towards developing one [8], [11].

Our approach is an attempt to design a system capable of representing the various legal modalities (ought, ought not, may, or can) and delivering pragmatic information for the users based on generalist and sometimes abstract body of laws. It must then by default be designed to integrate the fast evolution of the regulations, the divergent or particular controls as well as being able to focus on a firm specific information systems' environment. Finally, this system must furnish the necessary information to apprehend the business and operational impacts of regulatory requirements. Our ontology does not seek to assess the effective compliance nor the threat landscape of a company. Our work is solely to express the requirements and constraints based on the deontic models of the laws.

The complexity of DR management resides in the necessity of translating the regulatory constraints and requirements into technical, organizational and operational terms. Not to mention that DR is context specific and depends on one organization's markets, geographical presence and jurisdictions, it therefore requires an in-depth analysis involving a broad set of skills fragmented across the organization's departments. We then identified three main types of users which are: IT managers, security practitioners and compliance managers. All three of them require different pieces of information extracted from the laws in order to perform their duties while ensuring business continuity and their company compliance. For example, the IT manager will need the deontic modalities and regulatory requirements to build and manage the overall information systems while the security practitioners will focus on the mandatory security controls that need to be implemented.

III. ONTOLOGY BUILDING

A. Reused Ontology

During our search we were able to distinguish two main areas of work related to ours.

1) Information Security Management Ontologies

As show in [16], security ontologies can be sorted by: general security ontology, security ontology applied to a specific domain and theoretical work. This work was later reused by [17] who extended the classification to eight categories, namely: beginning security ontologies, security taxonomies, general security ontologies, specific security ontologies, web oriented security ontologies, risk based security ontologies, ontologies for security requirements and security modeling ontologies. They reached the conclusion that the existing security ontologies vary a lot and no ontology covers all of the aspects of the security domain.

A strong basis for information security domain knowledge may be found in [18]. Their Information Security Ontology is composed of three sub-ontologies (security, enterprise and location) and is based on established documentation, industry best practices and controls. In their previous work, [19] also proposed a security ontology as a basis for a low cost risk management solution as well as an ontology focusing on threats to corporate assets. The ontology consists of five sub-ontologies (threat, attribute, infrastructure, role and person). Other works introduce ontologies specific to vulnerability analysis and management [20], risk assessment [3], security annotations of agents and web services [21], dependability requirements that include security [22], secure development [23]. Despite the variety of domain specific ontologies in the different branches of information security, they tend to apply to only very limited scope which prevent us from reusing most of them. We will nonetheless reuse the role and person concepts found in [18] as much as possible.

2) Compliance & Legal Ontologies

Several conceptualizations of the legal domain have been presented or studied and comparison of legal ontologies can also be found [4, 5]. For instance, the McCarty's Language for Legal Discourse [24] is semi-formal conceptualization with the ambition of creating a general language for legal domain knowledge. By dividing the domain in three: norm, act and concept description, the issue of reusability of legal ontologies is presented in [25]. The three concepts are designed to be sufficient to conceptualize the subdomains of the legal domain.

There are also ontologies focusing on a single regulation or a type of regulation such as privacy ontologies. For instance, PrivOnto [26] is a semantic framework to represent annotated privacy policies and provide a linguistic instrument for the privacy domain. Another example is GDPRtEXT [27] which is a list of concepts present in the General Data Protection Regulation (GDPR). Its goal is to provide a way to refer to the concepts and terms found in the GDPR without providing an interpretation of compliance obligations. Similarly, the privacy ontology PrOnto [28], models the GDPR main conceptual cores "to support legal reasoning and compliance checking by employing defeasible logic theory". Similarly to the Frame-Based Ontology, PrOnto manages to model norms through its conceptualization of deontic operators. We will reuse and follow as much as possible these design patterns for our ontology.

Finally, LKIF [15] is a legal core ontology presented as a knowledge representation formalism that enables the translation between different legal bases. Comparably to the role and person concepts found in [26], LKIF presents the organization, role and person concepts which we will be reusing.

B. Ontology Capture

The preceding sections presented the requirements for our ontology and some concepts we reuse from existing ontologies.

1) Key Concepts

Capturing our ontology implies the findings of precise unambiguous text definitions and terms' identification for the different concepts and relationships [8]. We group the top level concepts of our ontology in four subontologies: enterprise, security, legal and location.

We reuse the top concepts Individual and Role from [15,18]. The concept Individual (Synset ID: 100007846), (ent: Individual ⊑ ⊤) is used to represent an identifiable natural person. The concept Role (Synset ID: 100722061), (ent: Role ⊑ ⊤) and its corresponding subconcepts are used to represent the normal or customary activity of a person in a particular social setting. Every individual has one or more roles which enables a flexible handling of the concepts in complex scenarios.

The creation of the subontologies enterprise, security and location is derived from [18]. While the whole subontologies do not fit the needs of ours, we reuse and adapt their concepts Control, Asset, Organization, Data and Location to create respectively Security_Measure (Synset ID: 100823316), (sec: Security_Measure ⊑ ⊤), Information_System (Synset ID: 103164344), (ent: Information_System ⊑ ⊤), Legal_Entity (Synset ID: 100001740), (ent: Legal_Entity ⊑ ⊤), Technological_Data (Synset ID: 105816622), (ent: Technological_Data ⊑ ⊤), Country (ent: Loc: Country⊑ ⊤), (Synset ID: 108544813) and Citizenship (loc: Citizenship ⊑ ⊤) (Synset ID: 113953467).

The concept Technological_Data and its corresponding subconcepts are used to represent data in digital format. For this ontology, we model the subconcepts: Business_Data and Personnal_Data. The former (ent: Business_Data ⊑ Technological_Data) corresponds to data involved in the course of conduct of activities of a Legal_Entity while the latter (ent: Personnal_Data ⊑ Technological_Data) are any information relating to an identified or identifiable natural person.

The concept Legal_Entity and its corresponding subconcepts represent a natural or legal person, a public authority body which carries out an activity whatever its legal form. The following subconcepts modeled so far are: Business_Organization, Independant_Organization and Regulatory_Agency.

We use the concept Information_System to describe an organized set of resources (hardware, software, individual, data and procedure) which makes it possible to process data.

Accordingly, we create the concept IT_System (Synset ID: 104377057), (ent: IT_System ⊑ ⊤) to represent a combination of interacting elements (resources) organized to achieve one or more desired objectives. We introduce this concept to provide an agile ontological structure according to the granularity of regulations. To illustrate various data processing, we add the concept Action (Synset ID: 100037396), (ent: Action ⊑ ⊤) and its corresponding subconcepts to represent something done (i.e. action or processing of data).

We then need to create the concept Functionnal_Process (SynSet ID: 101023820), (ent: Functionnal_Process ⊑ ⊤) to describe a set of interrelated or interacting activities that uses inputs to produce an intended result. As an example, an instance of a Functionnal_Process would be the payroll process within an organization.

Security measures are usually gathered within different classes of documents. We then create the concept Documentation (Synset ID: 106588326), (sec: Documentation ⊑ ⊤) to represent the set of documents such as policies, guidelines, procedures or frameworks. The concept is also useful to illustrate external documentation such as standards and frameworks which are often cited in regulations.

We need the concept Norm (Synset ID: 106532330), (leg: Norm ⊑ ⊤) to describe texts of laws. To show an action that is governed by a regulation through legal modalities, we will use the concept Act (Synset ID: 100030358), (leg: Act ⊑ ⊤) as introduced by [23]. Accordingly, a Norm governs an Act which itself governs Individual, Technological_Data, Legal_Entity and Security_Measure.

2) Key Relationships

Our next task focuses on determining the relationships between the concepts. Our model consists of two types of relationships: characteristic relationships which are used to represent the links between the different concepts of the model and action relationships when a concept performs a direct action on another concept. Our model is composed of 11 characteristic relationships (govern, has_a, isLocatedIn, belong, involve, protect, define, manage, isOwnedBy, isComposedOf and create) and 3 action relationships (process, isUsedBy, perform).

C. Formalization of the U.S. Export Arm Regulations

To illustrate the different primary concepts of our model, we will formalize parts of the Export Arm Regulations [29]: EARNorm is_a Norm. EAR Supplement No. 18 to part 734 states the following:

Transmitting or otherwise transferring "technology" or "software" to a person in the United States who is not a foreign person from another person in the United States.

Using the concept Act, Supplement No. 18 to part 734 is therefore represented as: EAR734.18Act is_a Act. Using the govern relation: EARNorm governs EAR734.18Act. Data regulated by the EAR Supplement No. 18 to part 734 then correspond to: EARBusiness_Data is_a (Business_Data ⊑ Technological_Data). We then need to create a first person using the concept Individual: PersonReceiveingEARData is_a Individual. Then, this individual must be physically located in the United States: US is_a Country and be an US citizen: USCitizenship is_a Citizenship. PersonReceiveingEARData isLocatedIn US and has_a USCitizenship. We can proceed to create our second individual residing in the US who is the sender of the data: PersonSendingEARData is_a Individual and isLocatedIn US.

Translating the term Release into practical terms would result in granting or receiving access to EAR controlled data. To encapsulate this, the concept Action will be used to represent the transfer of controlled data: TransferEARData is_a (Transfer ⊑ Action). To add an extra layer of granularity, we can come up with additional subconcepts such as granting access and its reverse, receiving access: GrantAccessEARData is_a (GrantAccess ⊑ Transfer) and ReceiveAccessEARData is_a (ReceiveAccess ⊑ Transfer). In the end, transmitting or otherwise transferring would be: An individual that uses an

EARSystem is_a IT_System to perform the action TransferEARData that process EARBusiness_Data.

EAR734.18Act ⊑ governs ((PersonReceiveingEARData ⊓ isLocatedIn.US ⊓ has_a.UScitizenship) and (EARSystem ⊓ perform.ReceiveAccessEARData ⊓ process.EARBusiness_Data))

EAR734.18Act ⊑ governs ((PersonSendingEARData ⊓ isLocatedIn.US) and (EARSystem ⊓ perform.GrantAccessEARData ⊓ process.EARBusiness_Data))

IV. Conclusion and Further Research Direction

Based on various data sources such as established documentation or industry best practices, existing ontologies [15, 18] and regulations, we present an ontology able to formalize firms' legal context while enabling the sharing and reuse of knowledge to support decision making. We present an ontology with 14 top level concepts grouped in four subontologies (enterprise, security, legal and location) and 14 relationships. With the ambition of facilitating the apprehension of business and operational impacts of regulatory requirements, our ontology is designed for any type of firm. We are currently developing the ontology using Protégé and implementing it at a worldwide chemical company subject multiple regulations.

We also plan to integrate further existing information security and risk management ontologies. We believe that combining them will enable more efficient risk management by combining regulatory risk and information security risk.

References

[1] Delorme, G., Talens, G., Disson, E., Collard, G., & Gaget, E. (2020, December). On the Definition of Data Regulation Risk. In International Conference on Service-Oriented Computing (pp. 433-443). Springer, Cham.

[2] Donner, M. (2003). Toward a security ontology. IEEE Security & Privacy, 1(03), 6-7.

[3] Tsoumas, B., & Gritzalis, D. (2006, April). Towards an ontology-based security management. In 20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06) (Vol. 1, pp. 985-992). IEEE.

[4] Larmande, P., Arnaud, E., Mougenot, I., Jonquet, C., Rouge, T. L., & Ruiz, M. (2013, May). Proceedings of the 1st International Workshop on Semantics for Biodiversity. In 1. International Workshop on Semantics for Biodiversity (pp. 001-131).

[5] Visser, P. R., & Bench-Capon, T. J. (1998). A comparison of four ontologies for the design of legal knowledge systems. Artificial Intelligence and Law, 6(1), 27-57.

[6] Jones, D., Bench-Capon, T., & Visser, P. (1998). Methodologies for ontology development.

[7] Gruber, T. R. (1992). Ontolingua: A mechanism to support portable ontologies.

[8] Uschold, M., & King, M. (1995). Towards a methodology for building ontologies (pp. 1-13). Edinburgh: Artificial Intelligence Applications Institute, University of Edinburgh.

[9] WordNet, https://wordnet.princeton.edu/ last accessed 2022/02/20.

[10] Fox, M.S., Chionglo, J., Fadel, F. A Common-Sense Model of the Enterprise, Proceedings of the Industrial Engineering Research Conference 1993

[11] Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering.

[12] Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., & García-Castro, R. (2022). LOT: An industrial oriented ontology engineering framework. Engineering Applications of Artificial Intelligence.

[13] Pinto, H. S., & Martins, J. P. (2004). Ontologies: How can they be built?. Knowledge and information systems, 6(4), 441-464.

[14] Niles, I., & Pease, A. (2003). Mapping WordNet to the SUMO ontology. In Proceedings of the ieee international knowledge engineering conference (pp. 23-26).

[15] Alexander, B. O. E. R. (2009). LKIF core: Principled ontology development for the legal domain. Law, ontologies and the semantic web: channelling the legal information flood, 188, 21.

[16] Blanco, C., Lasheras, J., Valencia-García, R., Fernández-Medina, E., Toval, A., & Piattini, M. (2008, March). A systematic review and comparison of security ontologies. In 2008 Third International Conference on Availability, Reliability and Security (pp. 813-820). Ieee.

[17] Souag, A., Salinesi, C., & Comyn-Wattiau, I. (2012, June). Ontologies for security requirements: A literature survey and classification. In International conference on advanced information systems engineering (pp. 61-69). Springer, Berlin, Heidelberg.

[18] Fenz, S., & Ekelhart, A. (2009, March). Formalizing information security knowledge. In Proceedings of the 4th international Symposium on information, Computer, and Communications Security (pp. 183-194).

[19] Ekelhart, A., Fenz, S., Klemen, M. D., & Weippl, E. R. (2006, December). Security ontology: Simulating threats to corporate assets. In International Conference on Information Systems Security (pp. 249-259). Springer, Berlin, Heidelberg.

[20] Wang, J. A., & Guo, M. (2009, April). OVM: an ontology for vulnerability management. In Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (pp. 1-4).

[21] Denker, G., Kagal, L., Finin, T., Paolucci, M., & Sycara, K. (2003, October). Security for daml web services: Annotation and matchmaking. In International Semantic Web Conference (pp. 335-350). Springer, Berlin, Heidelberg.

[22] Dobson, G., & Sawyer, P. (2006, November). Revisiting ontology-based requirements engineering in the age of the semantic web. In Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs (pp. 27-29).

[23] Karyda, M., Balopoulos, T., Dritsas, S., Gymnopoulos, L., Kokolakis, S., Lambrinoudakis, C., & Gritzalis, S. (2006, April). An ontology for secure e-government applications. In First International Conference on Availability, Reliability and Security (ARES'06) (pp. 5-pp). IEEE.

[24] McCarty, L. T. (1989, May). A language for legal discourse i. basic features. In Proceedings of the 2nd international conference on Artificial intelligence and law (pp. 180-189).

[25] Van Kralingen, R. (1997, June). A conceptual frame-based ontology for the law. In Proceedings of the first international workshop on legal ontologies (pp. 6-17).

[26] Oltramari, A., Piraviperumal, D., Schaub, F., et al., (2018). PrivOnto: A semantic framework for the analysis of privacy policies. Semantic Web, 9(2), 185-203.

[27] Pandit, H. J., Fatema, K., O'Sullivan, D., & Lewis, D. (2018, June). GDPRtEXT-GDPR as a linked data resource. In European Semantic Web Conference (pp. 481-495). Springer, Cham.

[28] Palmirani, M., Martoni, M., Rossi, A., Bartolini, C., & Robaldo, L. (2018, October). Pronto: Privacy ontology for legal compliance. In Proc. 18th Eur. Conf. Digital Government (ECDG) (pp. 142-151).

[29] Export Administration Regulation (EAR), 15 C.F.R. § 730 et seq, https://www.bis.doc.gov/index.php/regulations/export-administration-regulations-ear, last accessed 2022/02/20.

SMIFIER: A Smart Contract Verifier for Composite Transactions

Yu Dong*, Yue Li*, Dongqi Cui†, Jianbo Gao*, Zhi Guan‡, and Zhong Chen*
*School of Computer Science, Peking University, China
†National Engineering Research Center for Software Engineering, Peking University, China
Email: *{dongyu1101, liyue_cs, gaojianbo, zhongchen}@pku.edu.cn, †cdq@stu.pku.edu.cn, ‡guan@pku.edu.cn,

***Abstract*—Ensuring functional correctness of smart contracts is a pressing security concern to blockchain-based systems. With the development of blockchain application, the trading scenarios and function implementation of smart contracts have become increasing complex, containing several interacted contracts or related functions. However, the existing contracts verifiers for proving functional correctness focus on verifying isolated contract or function but ignore the interactions between them, which makes it difficult to verify correctness of *composite transactions*, i.e., complex transaction scenarios that invoke multiple contracts or trigger a set of transactions. In this paper, we present SMIFIER, a formal verification tool for smart contracts to prove functional properties in composite transactions. SMIFIER defines a set of specifications for composite transactions and can automatically specify properties in these multiple complex transactions. Based on states extraction and mapping, SMIFIER translates annotated Solidity program into Boogie program and verifies relations between functions and properties for interacted contracts. Our experimental evaluation on 12 real-world projects and 65 properties, demonstrates that SMIFIER is practically effective in ensuring functional correctness of properties in composite transactions.**

***Index Terms*—Smart Contract, Formal Verification, Composite Transaction**

I. INTRODUCTION

Ensuring correctness of smart contracts, programs that stored and executed on the blockchain, is an urgent security concern. Smart contracts always store and manage millions of monetary assets, making their security highly sensitive. The pressing need for contract correctness has gained via several recent high-value incidents resulting in massive losses [1]. Unfortunately, most existing security analyzers [2]–[4] focus on universal security vulnerabilities such as contract reentrancy and integer overflow, but ignore functional correctness of contracts, i.e., the functional obligations that the program is intended to implement. In practice, the functionality of smart contracts implies transaction logic and supports a variety of blockchain applications. If developers make mistakes and implement functionality incompletely, the contracts will not behave as intended and will expose vulnerabilities with potentially devastating financial effects. Therefore, it is critical to prove functional correctness of smart contracts.

With the development of blockchain applications, the function implementation of smart contracts has become increasing complex, containing the combination of interacted contracts or transactions. The implementation of a trading scenario always means exploiting a transaction sequence, which involves several interacted contracts externally calling each other, or successive transactions sent to related functions of a contract. As for the contracts in such *composite transactions*, code might appear secure, yet expose vulnerabilities when it interacts with other code. Existing security analyzers always ensure functional correctness of code in isolation such as single contract or function, but cannot verify an entire transaction sequence executing as intended.

Checking for functional correctness of smart contracts in composite transactions should overcome these challenges: 1) composite transactions contain multiple transaction rules and specific contract functionality, thus it is difficult to give a general and explicit definition of composite transactions. Also, to specify properties in these transactions, it is essential to design multiple specifications to describe relations of functions and interactions of contracts. 2) to verify the interaction of contracts, we must check the behaviors of called contracts after invocation. Unfortunately, it is difficult to trace and explore states of called contracts.

In this work, we introduce SMIFIER, an automated formal verifier for proving functional correctness of smart contracts in composite transactions. To achieve this, we give an explicit definition of composite transaction and define multiple kinds of specifications for it. Based on translating Solidity [5] program into an intermediate verification language Boogie [6], SMIFIER can specify function relations and interacted contract properties. In particular, SMIFIER takes advantage of well-engineered pipeline and automated verification conditions generator of Boogie. We describe SMIFIER through examples and evaluate SMIFIER on real-world contracts in composite transactions. We also compare SMIFIER with state-of-the-art safety analysis tools and demonstrate SMIFIER greatly outperforms other tools in terms of specifying properties in composite transactions.

The contribution of our work is as follows:

- We give a definition of composite transaction which can provide an explicit description of multiple complex trading scenarios involving several interacted functions or contracts.
- We impose a framework called SMIFIER, which can automatically verify functional correctness of contracts in composite transactions. For conveniently specifying properties in interacted functions or contracts, we design a set of specifications applied for composite transactions.

DOI reference number: 10.18293/SEKE2022-128

```
1  /// @notice invariant enableTransfer ==> transferFrom
2  contract DaiToken {
3    mapping(address => uint) public balances;
4    address public owner;
5    uint public totalSupply;
6    bool public transferEnabled;
7
8    constructor() public {
9      owner = msg.sender;
10     balances[msg.sender] = totalSupply;
11   }
12   function enableTransfer() public {
13     transferEnabled = true;
14   }
15   function transferFrom(address _from, address _to, uint
         _value) public {
16     require(transferEnabled == true);
17     balances[_from] -= _value;
18     balances[_to] += _value
19   }
20 }
```

```
21 /// @notice invariant sum(DaiToken.balances) == DaiToken.
       totalSupply
22 contract TokenFarm {
23   DaiToken public daiToken;
24
25   constructor(DaiToken _daiToken) public {
26     daiToken = _daiToken;
27   }
28
29   /// @notice precondition DaiToken.transferEnabled ==
         true
30   function deposit(uint _amount) public {
31     daiToken.transferFrom(msg.sender, address(this),
           _amount);
32     _mint(msg.sender, _amount);
33   }
34 }
```

Fig. 1: Simplified Token Farm source code.

- We implement the framework and demonstrate the validation of our tool. We perform the evaluation over 12 real-world projects and 65 properties defined for composite transactions, showing that SMIFIER can effectively verify properties for smart contracts in composite transactions.

II. MOTIVATING EXAMPLE

In this section, we present a motivating example of contracts in composite transaction. Fig. 1 presents the simplified source code from Token Farm, a common DeFi (Decentralised Finance) application [7]. Through Token Farm, users can deposit Dai tokens (a stablecoin cryptocurrency) to the contract and it will mint and transfer Farm Tokens (cryptocurrency provided by the application) to them. The users can later withdraw their Dai tokens by burning their Farm Token on smart contract and the Dai tokens will be transferred back to them. As mentioned, the annotations (beginning with "`/// @notice`") in code will be illustrated in the following section.

In contract DaiToken, the function `enableTransfer` sets the state variable `transferEnabled` to true (line 13 in Fig. 1). Only after the owner makes transfers enabled, the function `transferFrom` can transfer the specified amount of tokens from the address `_from` to the address `_to` (line 15-19). The successive transactions sent to function `enableTransfer` and function `transferFrom` make up a transaction sequence. If the developer forgets writing the `require` clause (line 16), the functionality that tokens are tradeable only after the operation of owner will be incomplete. However, the result of vulnerable `transferFrom` would be identical to the correct one when verifying the single function. Therefore, the proving of this functionality can only be achieved by putting the two transactions together to validate whether the entire transaction sequence is executed in order.

In contract TokenFarm, function `deposit` externally calls function `transferFrom` to transfer several Dai tokens from msg.sender to this address (line 31). Contract DaiToken is a token contract and implements standard interfaces. The developers always assume the implementation is consistent with the standard interfaces and satisfies intended functionality. However, if the token contract makes fake implementation by only invoking standard interfaces without correct code writing, developers will mistakenly call the false function. For example, if `transferFrom` deliberately mistakenly writes the plus and minus operators by writing `balances[_from] += _value` and `balances[_to] -= _value`, "deposit" will become "withdraw" when TokenFarm calls this wrong function, which makes the meaning completely opposite and may lead to unexpected financial loss. Therefor, it is essential to verify functional correctness of the called contract before sending message calls.

III. SMIFIER

In this section, we will describe the definition of composite transaction and the details of SMIFIER in verifying smart contracts. Our verification architecture is summarized in Fig. 2, which consists of three phases. In the first phase, SMIFIER parses annotated smart contracts and generates an abstract syntax tree (AST). In the second phase, SMIFIER traverses the AST and converts Solidity program into Boogie by modular program reasoning. For specifications for composite transactions, we instrument the Boogie program by states recognition, extraction and mapping. Finally, SMIFIER relies on Boogie verifier to prove correctness or reports the violated annotations in Solidity program.

A. Definition of Composite Transactions

Composite transaction means a trading scenario consisting of multiple transactions involving several related functions in a contract or several interacted contracts. We define two kinds of transaction sequences as composite transaction, one *inter-txn* and another *intra-txn*.

Inter-txn. Inter-txn means a transaction sequence containing successive transactions sent to related functions in a contract. Inter-txn is operated by the message sender who sends transactions to contract. We let S represent a transaction sequence,

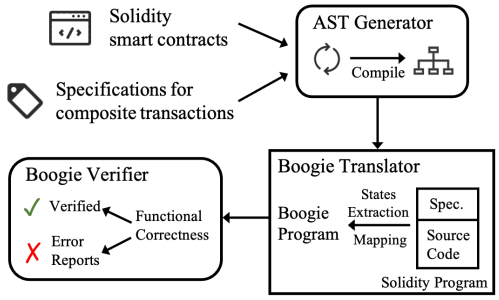


Fig. 2: Schematic workflow of SMIFIER.

TABLE I: Specifications for inter-txn and intra-txn. The first two are for inter-txn while the last one is for intra-txn.

Property	Notation	Description
Precedence Relation	$func1 \Longrightarrow func2$	$func2$ requires the variable var is equal to the value set by $func1$. Insert as contract-level invariant.
Exclusion Relation	$func1 \,\|\|\, func2$	$func2$ requires the variable var is not equal to the value set by $func1$. Insert as contract-level invariant.
Other Contract Property	$Spec\,\{C.var\}$	Specify state variable var of contract C. Insert as contract-level invariant, function-level pre- and post-conditions.

$C.f$ represent invoking function f in contract C. The inter-txn can be expressed as $S = \langle C.f_1, C.f_2, C.f_3, \cdots \rangle$ where f_1, f_2, f_3 have relation r. We define r as $f_1 \otimes_v f_2 = f_2 \rightarrow (f_1 \sim v)$, where two functions refer to the same state variable v and the requirement in f_2 depends on the modification in f_1. We divide the relations into two types, **precedence relation** and **exclusion relation**, respectively expressed as $f_1 \otimes_v f_2 = f_2 \stackrel{=}{\rightarrow} (f_1 \sim v)$ and $f_1 \otimes_v f_2 = f_2 \stackrel{\neq}{\rightarrow} (f_1 \sim v)$. Precedence relation requires v is equal to the value set by f_2, while exclusion relation requires not equal to. The error handling statements(e.g., assert, require) in f_2 ensures f_2 is executed after f_1 was invoked. For example, function `enableTransfer` (in Fig. 1) modifies `transferEnabled` to true and the execution of `transferFrom` requires the modified value is equal to true. We say the transactions sent to these two functions in sequence form the inter-txn and two functions have precedence relation.

Intra-txn. Intra-txn means a transaction sequence involving a set of contracts which externally call each other. Intrax-txn is operated by the program developer. The intra-txn can be expressed as $S = \langle C_1.f_1, C_2.f_2, C_3.f_3, \cdots \rangle$ where C_1, C_2, C_3 are different contracts and $C_1.f_1$ calls $C_2.f_2$ and $C_2.f_2$ calls $C_3.f_3$. For example, the function `deposit` in contract TokenFarm externally calls the function `transferFrom` in contract DaiToken (in Fig. 1). The transaction sent to contract TokenFarm and then exposing message calls to another contract DaiToken forms the intra-txn.

B. Specifications for Composite Transactions

We design multiple specifications so that SMIFIER could verify function relations for inter-txn and called contract properties for intra-txn, as shown in Table. I. To be mentioned, the specifications are inserted as in-code annotations supported by Solidity and consist of three kinds of statements, *pre- and post-conditions* and *invariants*. Contract-level invariants must hold before and after the execution of every public function. Functions are specified with pre- and post-conditions, which hold before entering functions and specify final states of functions.

For inter-txn, we design two kinds of specifications respectively for two function relations. We let "$\Longrightarrow$" represent precedence relation and "||" for exclusion relation. $func1 \Longrightarrow func2$ means $func2$ should be called after $func1$ was invoked because $func2$ requires var is equal to the value assigned in $func1$ while $func1 \,||\, func2$ means $func1$ and $func2$ cannot be called simultaneously. For example, the precedence relation in Fig. 1 is expressed as annotation "`enableTransfer` $\Longrightarrow$ `transferFrom`" (line 1).

For intra-txn, we create a new format $Spec\,\{C.var\}$. $Spec$ contains all provided forms of property expressions, as well as the specifications designed for inter-txn described above. Different with message calls to contracts supported by Solidity, SMIFIER uses $C.var$ to represent the variable of called contract, where C is the called contract name rather than instance name and var is the variable belonging to C. Intra-txn specifications are inserted in the current contract as all three kinds of specification statements. For example, we insert contract-level invariant in contract TokenFarm (line 21 in Fig. 1) to ensure the sum of individual balance is equal to the total supply of DaiToken. The special function `sum` is provided to express the sum of collections (arrays and mappings). Besides, we can also insert pre-condition before where the call happens to verify the function `enableTransfer` (line 29).

C. Transform and Verification

We implement a transform algorithm for translating Solidity to Boogie program in composite transactions. As shown in Algorithm. 1, given a program P as a list of contracts source codes and S as a list of specifications, the algorithm produces an Boogie program $P^{'}$ that inserted with transformed annotations. The algorithm has three parts. Line 1 transfers the Solidity program into Boogie program without specifications. Lines 3-16 handles the inter-txn specifications in S, while lines 17-27 handles the intra-txn specifications in S.

For each specification s in S, the algorithm first identifies the type of the specification. If s contains the function relation operators "$\Longrightarrow$" or "||", we predicate s is for inter-txn and define the two functions before and after the relation operator as f_1 and f_2. Otherwise we identify s is for intra-txn if s contains the name of another contract. For inter-txn, the algorithm extracts requirement in f_2 and assignment in f_1, and then respectively converts into pre- and post-condition. To verify the priority and exclusion of execution, the algorithm generates a new procedure $proc_1$ calling two functions in order and declares another procedure $proc_2$ calling only f_2 if we verify precedence relation. If $proc_1$ is proved and $proc_2$ fails verification, we indicate that f_2 cannot be executed alone and

Algorithm 1: Transform algorithm

Input: Solidity program P as a list of contracts source codes and S as a list of specifications
Output: Transformed Boogie program $P^{'}$

Transfer Solidity program P to Boogie program $P^{'}$ without S
for $s \in S$ **do**
  if s *contains* "$\Longrightarrow$" *or* "||" **then**
    $r \leftarrow$ "$\Longrightarrow$" or "||"
    $f_1 \leftarrow$ Function before r in s
    $f_2 \leftarrow$ Function after r in s
    $pre \leftarrow$ Rewrite $require$ statement in f_2
    Insert pre before f_2 into $P^{'}$
    $post \leftarrow$ Transfer $assignment$ statements in f_1
    Insert $post$ after f_1 into $P^{'}$
    Declare a new procedure $proc_1$ in $P^{'}$
    $proc_1 \leftarrow$ "call f_1; call f_2;"
    if r *is* "$\Longrightarrow$" **then**
      Declare a new procedure $proc_2$ in $P^{'}$
      $proc_2 \leftarrow$ "call f_2;"
    end
    Insert $proc_1$ (and $proc_2$) into $P^{'}$
  end
  else if s *contains another contract name* **then**
    $C \leftarrow$ The current contract
    $C^{'} \leftarrow$ The called contract
    $vars \leftarrow$ The variables names of $C^{'}$ in s
    $func \leftarrow$ The called function of $C^{'}$
    Create a new specification $s^{'}$ and copy s to $s^{'}$
    for var *in* $vars$ **do**
      $bg\_var \leftarrow$ Locate Boogie variable in $func$ in $P^{'}$ according to type and name of var
      Replace $C.var$ in $s^{'}$ with bg_var
    end
    Insert $s^{'}$ in $P^{'}$ in the same position as s in P
  end
  else
    Transfer and insert s in $P^{'}$
  end
end
Return $P^{'}$

the invocation of f_2 requires f_1, i.e., the precedence relation property is verified. Also, we predicate exclusion relation is proved when the procedure calling two functions fails verification, as two functions cannot execute simultaneously.

For intra-txn, the algorithm locates the called variable names of called contract and replaces them with the translated Boogie representation. We identify variables in Boogie program by first determining whether they are state variables or local variables and then traversing the global or local variable list (of called function) to search them. After research, we replace $C.var$ with variables in Boogie and insert the replaced specification in front of the called contract or function and keep the specification types the same.

Fig. 3 presents the simplified transformed Boogie program of Token Farm contracts (Fig. 1). According to our algorithm, we create two procedures `proc1` and `proc2` in DaiToken respectively calling two functions and only the latter function. The new procedure parameters are made up of arguments of functions called in the procedure. For intra-txn, we move specifications (previously inserted in contract TokenFarm) to the front of called contract DaiToken and called function `transferFrom`. We also replace variables in properties (e.g., `DaiToken.balances`) with Boogie expressions.

```
1  // invariant sum(balances) == totalSupply
2  contract DaiToken {
3    var balances: [address]int;
4    var transferEnabled: bool;
5    procedure enableTransfer(...) { ... }
6    // precondition transferEnabled == true
7    procedure transferFrom(_from: address, _to: address,
         _value: uint) { ... }
8    procedure proc1(_from: address, _to: address, _value:
         uint) {
9      call enableTransfer();
10     call transferFrom(_from, _to, _value);
11   }
12   procedure proc2(_from: address, _to: address, _value:
         uint) {
13     call transferFrom(_from, _to, _value);
14   }
15 }
16 contract TokenFarm {
17   DaiToken public daiToken;
18   procedure deposit(_amount: uint) {
19     // call transferFrom() in DaiToken
20   }
21 }
```

Fig. 3: Simplified transformed Boogie program of Token Farm.

After being transferred to Boogie program, SMIFIER leverages Boogie verifier to transform the program into verification conditions and discharge them using SMT solvers. SMIFIER verifies each procedure of each contract and outputs the verification result of each procedure. If Boogie proves the correctness of program, the result will display "OK". Otherwise if there are vulnerabilities in program, SMIFIER will report "Error" and map the violated annotations back to the Solidity code (e.g., line numbers, function names).

IV. EVALUATION

We now present out evaluation of SMIFIER on real-world Ethereum projects. We focus on the following key questions: 1) What types of properties are common for composite transactions in real world? 2) How effective is SMIFIER in verifying smart contracts in composite transactions? 3) How does SMIFIER compare with other smart contract safety analysis tools? All experimental results reported in this section are conducted on a server running Ubuntu 20.04 LTS with 32 AMD EPYC CPUs at 2.8GHz and 64GB of physical memory.

A. Benchmark and Properties

Benchmark. We have collected in total 12 Ethereum projects in ERC-20, ERC-721 standards and DeFi applications. We focus on these projects because 1) they are most widely used contracts, 2) they contain several related functions in a contract or several interacted contracts which satisfy definition of composite transactions, 3) we want to focus our analysis on contracts that manipulate critical digital assets. Our benchmark includes the top five ERC-20 contracts and the top five ERC-721 contracts ranked by Etherscan [8] which define the interface and specification for implementing fungible and non-fungible tokens respectively. We also include two decentralized exchanges (DEX), Uniswap and Sushiswap, which

TABLE II: Properties for composite transactions and concrete examples taken from the benchmarks.

Type	Description	Example
User-based access control	Only particular users have privileges to perform critical actions.	`function setOwner() {owner = msg.sender;}` `function withdraw() {require(msg.sender == owner);}`
Token-based circulation control	The circulation of token is allowed at certain states.	`function enableTransfer() {transferEnabled = true;}` `function transferFrom() {require(transferEnabled == true);}`
Contract life circle	The transaction of contract is allowed at certain states.	`function pause() {paused = true;}` `function unpause() {require(paused == true);}`
Balances consistence	The sum of individual balance keeps invariant.	`invariant sum(ERC20.balance) == ERC20.totalSupply`
State-based properties	Defines states which invariants must hold or variables must satisfy.	`invariant ERC721._approve ==> ERC721.transferFrom`

use decentralized network protocols to facilitate automated transactions between cryptocurrency tokens.

Properties. To focus on verifying composite transactions, we have summarized common properties and classified them into five distinct categories, as shown in Table. II.

1) *User-based access control* defines only particular users have privileges to perform critical actions. This property is expressed as function relation, where a user has permissions to invoke another function only after being granted access in one function. The example in Table. II, taken from CK contract, stipulates that only the owner set in function `setOwner` can invoke `withdraw`.
2) *Token-based circulation control* means that the circulation (minting, release and transaction) of token is allowed at certain states. Only after the tokens are set accessible in one function, the manipulation of tokens in another function can be invoked. The example in Table. II, taken from LEO contract, shows that function `transferFrom` can be invoked only after `enableTransfer` enables transfer.
3) *Contract life circle* defines at which states the transactions of contracts are allowed. The contracts usually have emergency stop mechanism or can be deprecated, controlled by state variables. The example in Table. II, taken from USDT contract, states that function `pause` triggers stopped state of the contract and the function `unpause` returns the contract to normal state.
4) *Balances consistency* indicates that the sum of individual balance keeps invariant after transfers occur. This property is applied to verifying functional correctness of the called contract. The example in Table. II ensures the sum of account balances is equal to the total supply in Uniswap contract.
5) *State-based properties* defines states which invariants must hold and variables must satisfy. This property can be inserted in single contract or function. It can also be expressed in the current contract as specification for the called contract. The example in Table. II shows the verification of precedence relation in MCHH contract.

B. Verifying Contracts using SMIFIER

We now report on the effectiveness of SMIFIER in verifying our benchmarks. For each project, we manually insert specifications using the representation defined in Section.III, which contain all kinds of properties described above. We represent out results in Table. III.

The key result is that SMIFIER can successfully verify 60 of the 65 properties (92.3%) in the benchmarks. The reason why SMIFIER cannot verify the remaining five properties is that the specification we designed is difficult to express some properties in ERC-721 contracts. There are several functions in ERC-721 depending on parameters but not state variables. If we specify parameters in properties, we are unable to extract them from functions as state variables when we implement transform algorithm. We will solve this problem in future work. The average verification time of SMIFIER is 3.72 seconds and the time is related to the lines of code. In general, the longer the program, the more contracts and functions the program contains, the more time SMIFIER takes to verify it. Because SMIFIER transforms and verifies each function of each contract in sequence.

Finally, we compare SMIFIER to state-of-the-art smart contract analysis tools: KEVM [9], SOLC-VERIFY [10], VERISOL [11] and VERX [12], as shown in Table. IV. Our benchmark consists the first four types of properties defined in the Table. II which are all for composite transactions, and the last property for single contract and function. Results indicate that other tools can only analyze properties in single function or contract or some properties about function relations while SMIFIER can verify other contract properties (balances consistence), which is very important for verifying complete functional correctness.

V. RELATED WORK

In recent years, there has been great interest in formally verifying the correctness of smart contracts. For instance, KEVM [9] translates EVM bytecode to KEVM and leverages the K framework [13] for checking contracts against given specifications while Ahrendt [14] translates Solidity into Java and uses KeY [15], a deductive Java verification tool. SOLC-VERIFY [10] and VERISOL [11] are two verifiers require users to manually provide annotations and check contracts based on translation to Boogie. Except formal verification, there are other systems using symbolic execution approach for verifying properties. VERX [12] can automatically prove temporal safety properties of smart contracts since it extracts predicates

TABLE III: Experimental results of SMIFIER verification on benchmarks. **LOC**: the number of lines of code, **Contracts**: the number of contracts in the program, **Functions**: the number of functions in the program, **Properties**: the number of properties we verified, **Verified**: the number of properties that were successfully proved, **Avg.time**: the average analysis time in seconds.

Contract	LOC	Contracts	Functions	Properties	Verified	Avg.Time (s)
USDT	447	7	25	4	4	2.46
CRO	641	8	30	5	5	2.87
WBTC	663	11	32	5	5	2.89
LEO	734	7	36	6	6	3.03
Fantom	624	6	46	7	7	2.92
LAND	1118	6	62	7	6	4.24
CK	1977	14	82	9	7	5.33
AXIE	891	7	53	4	4	3.24
MCHH	1192	10	56	6	5	4.21
Sherbet	1407	6	53	3	2	4.57
Uniswap	1256	7	52	5	5	4.46
Sushiswap	1353	8	47	4	4	4.42
Overall	12303	97	574	65	60	3.72

TABLE IV: Comparison of SMIFIER with other analysis tools. "✓" represents the tool can verify the property while "✗" represents the tool cannot verify the property.

Benchmark	KEVM	SOLC-VERIFY	VERISOL	VERX	SMIFIER
Single Contract Property	✓	✓	✓	✓	✓
Single Function Property	✓	✓	✓	✓	✓
User-based Access Control	✗	✗	✓	✓	✓
Token-based Circulation Control	✗	✗	✗	✓	✓
Contract Life Circle	✗	✗	✗	✓	✓
Balances Consistence	✗	✗	✗	✗	✓

automatically from the contract's source code. SMARTPULSE [16] models the contract's execution environment and uses CFGAR-based approach to check liveness properties.

VI. CONCLUSIONS

We presented SMIFIER, the first formal verification tool of Solidity smart contracts for composite transactions. This paper gives a definition of composite transaction which involves several related functions or interacted contracts and defines the two types it contains. SMIFIER presents a set of specifications and transform algorithm which specify properties and transfer Solidity program to Boogie program. After transform, SMIFIER relies on Boogie verifier to discharge verification conditions and prove function correctness. We demonstrated that SMIFIER is effective in specifying properties in composite transactions and proving functional correctness over 12 real-world Ethereum projects and 65 properties.

VII. ACKNOWLEDGEMENT

Zhi Guan is the corresponding author. Zhi Guan is supported by National Key R&D Program of China (NO.2020YFB1005800) and Beijing Natural Science Foundation(M21040).

REFERENCES

[1] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *International conference on principles of security and trust*. Springer, 2017, pp. 164–186.

[2] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 254–269.

[3] B. Mueller, "Smashing ethereum smart contracts for fun and real profit," *HITB SECCONF Amsterdam*, vol. 9, p. 54, 2018.

[4] M. Mossberg, F. Manzano, E. Hennenfent, A. Groce, G. Grieco, J. Feist, T. Brunson, and A. Dinaburg, "Manticore: A user-friendly symbolic execution framework for binaries and smart contracts," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 1186–1189.

[5] "Solidity documentation," 2022, https://docs.soliditylang.org/en/v0.8.12/.

[6] K. R. M. Leino, "This is boogie 2," *manuscript KRML*, vol. 178, no. 131, p. 9, 2008.

[7] "Decentralized finance," 2017, https://ethereum.org/en/defi/.

[8] "Ethereum (eth) blockchain explorer," 2015, https://etherscan.io/.

[9] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu *et al.*, "Kevm: A complete formal semantics of the ethereum virtual machine," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 204–217.

[10] Á. Hajdu and D. Jovanović, "solc-verify: A modular verifier for solidity smart contracts," in *Working Conference on Verified Software: Theories, Tools, and Experiments*. Springer, 2019, pp. 161–179.

[11] Y. Wang, S. K. Lahiri, S. Chen, R. Pan, I. Dillig, C. Born, I. Naseer, and K. Ferles, "Formal verification of workflow policies for smart contracts in azure blockchain," in *Working Conference on Verified Software: Theories, Tools, and Experiments*. Springer, 2019, pp. 87–106.

[12] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, and M. Vechev, "Verx: Safety verification of smart contracts," in *2020 IEEE symposium on security and privacy (SP)*. IEEE, 2020, pp. 1661–1677.

[13] G. Roşu and T. F. Şerbănuţă, "An overview of the k semantic framework," *The Journal of Logic and Algebraic Programming*, vol. 79, no. 6, pp. 397–434, 2010.

[14] W. Ahrendt, R. Bubel, J. Ellul, G. J. Pace, R. Pardo, V. Rebiscoul, and G. Schneider, "Verification of smart contract business logic," in *International Conference on Fundamentals of Software Engineering*. Springer, 2019, pp. 228–243.

[15] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, "Deductive software verification-the key book," *Lecture notes in computer science*, vol. 10001, 2016.

[16] J. Stephens, K. Ferles, B. Mariano, S. Lahiri, and I. Dillig, "Smartpulse: automated checking of temporal properties in smart contracts," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 555–571.

Ethereum Smart Contract Representation Learning for Robust Bytecode-Level Similarity Detection

Zhenzhou Tian[1,2,3*], Yaqian Huang[1,2], Jie Tian[1,2], Zhongmin Wang[1,2], Yanping Chen[1,2], and Lingwei Chen[4*]
[1]*School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an, 710121, China*
[2]*Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an, 710121, China*
[3]*Xi'an Key Laboratory of Big Data and Intelligent Computing, Xi'an 710121, China*
[4]*Department of Computer Science and Engineering, Wright State University, Dayton, OH, USA*

***Abstract*—Smart contracts are programs that run on a blockchain, where Ethereum is one of the most popular ones supporting them. Due to the fact that they are immutable, it is essential to design smart contracts bug-free before they are deployed. However, various defects have been found in the deployed smart contracts, causing huge economic losses and lowing people's trust. Writing secure smart contracts is far from trivial, where developers tend to engage in reliable resources or social coding platforms to reuse code. This leads to a large number of similar contracts with potential security risks. Therefore, detecting similarity of smart contracts helps to avoid vulnerabilities, identify threats, and improve the security of Ethereum. In this paper, we design a learning-effective and cost-efficient model, called SmartSD, for Ethereum smart contract similarity detection. Different from the current research efforts, SmartSD is performed on a bytecode level and leverages deep neural networks to learn the latent representations from the opcode sequences for smart contract bytecodes, where the representation learning and similarity measurement are supervised via siamese neural networks. The experimental evaluations demonstrate that SmartSD outperforms EClone's 93.27% accuracy, achieving 98.37% high detection accuracy and 0.9850 F1-score, which is computationally tractable and effectively mitigates the interference caused by compilers.**

***Index Terms*—Smart contract, Similarity detection, Deep learning, Siamese neural network**

I. Introduction

[1] Smart contracts are Turing-complete programs that run on a blockchain. Among the emerging blockchain platforms, Ethereum [27] is the most popular one to operate smart contracts. Different from the traditional programs, smart contracts have the unique characteristics of openness, transparency, non-tamperability, and independence from third parties. In other words, they can be integrated to different decentralized applications (e.g., financial services and supply chain management) after deployment; more importantly, they are immutable to be modified or patched, where the only way to change a smart contract is to deploy a new instance. It is hence essential to design smart contracts bug-free before their deployments [2]. Unfortunately, various defects have been found in the deployed smart contracts [3], [11], [25]. To put it into perspective, according to a recent study, 97% of the collected contracts are annotated as vulnerable by one or more vulnerability analysis tools [5]. This enables attackers to exploit the security vulnerabilities raised by these defects to fulfill their economic intents, and cause huge economic losses [19], which has also severely lowered people's trust in Ethereum smart contracts.

Smart contracts are usually written in high-level programming languages, where Solidity [20] is the most widely used one. With high-level languages facilitating smart contract developments, writing secure smart contracts is still not easy. The major factor behind this is that these programming languages are young and evolving, where every change of smart contracts would introduce significant updates into the languages. Developers may need many efforts to get familiar with newer versions, and thus tend to frequently engage in reliable resources (e.g., Etherscan) or social coding platforms (e.g., GitHub) for code reuse to expedite smart code development process. Considering the widespread defects existing in the deployed smart contracts, this naturally leads to a large number of similar smart contract codes with potential security risks. With this in mind, detecting the similarity of smart contracts may allow developers to avoid using the compromised ones; by comparing contracts with known vulnerabilities, it helps to identify threats and improve Ethereum security.

Similarity detection is a long-term research topic, but the investigation into smart contracts has been scarce, especially for similarity detection on Ethereum smart contract bytecodes. The deployment of a smart contract proceeds by first compiling its source code into EVM bytecode, encapsulating bytecode into a transaction, and then sending the transaction to the zero address [26]. On one hand, smart contract source codes are not always accessible to avoid attacks, while only their bytecodes are deployed to run on EVM. On the other hand, many smart contract defects occur after being compiled as bytecodes [21]. To this end, our research goal here is to automatically identify the similarities over the Ethereum smart contract bytecodes. To achieve this goal, we face a challenge: developers may compile smart contracts using different compilers or the same compiler with different optimization options enabled; accordingly, these cross-compiler and cross-optimization-level compilations will impose discrepancies on opcode distributions of the resulting bytecodes, even when their source codes are exactly the same. Direct fingerprint generation on bytecodes may not be a good idea to capture their semantics. It needs a better formulation to learn the

* Corresponding: tianzhenzhou@xupt.edu.cn and lgchen@mix.wvu.edu
[1]DOI reference number: 10.18293/SEKE2022-040

higher-level representations of contracts and automatically detect the similarity among them.

To address the above challenge, in this paper, we design a learning-effective and cost-efficient model, called *SmartSD*, for Ethereum **Smart** contract **S**imilarity **D**etection. Different from the limited research efforts that build upon either fingerprints susceptible to interference [10], cost-expensive runtime traces [14], [15], or source code structure [6], our model first decomposes the given bytecode into EVM instructions and abstract them as an opcode sequence, and then elaborate deep neural network structure to learn the representation encoding structures and semantics of the opcode sequence. In order to supervise the representation learning and similarity measurement, SmartSD further builds up siamese neural networks (SNN) [12] to train the model, such that the model can be successfully applied to the validation and test data sets. The major traits of our work can be summarized as follows:

- We investigate smart contract similarity detection on the bytecode level, leveraging feature learning ability of deep neural networks to learn latent representation from the abstracted opcode sequence for each bytecode. The proposed method is not only automatic and computationally tractable, but also effectively to mitigate the interference caused by compilers and their optimization options.
- We supervise representation learning and similarity measurement via SNN, and use the trained networks to infer the similarities of smart contract pairs.
- We formulate our positive and negative sample pairs from smart contract collection using Etherscan, and conduct extensive experimental evaluations on them, which demonstrate the robustness and effectiveness of SmartSD on smart contract similarity detection.

II. Motivation and Problem Statement

In this paper, we investigate Ethereum smart contracts developed in the high-level language Solidity. Due to the significant differences introduced by different major Solidity versions, the differences existing in smart contract bytecodes compiled by different Solidity compiler versions may be also significant. To understand such differences, we empirically compile a smart contract's source code using two different compiler version with and without optimization options enabled. Accordingly, we observe that the opcodes generated across these compilers are different regarding opcode numbers, opcode types, and the occurring frequencies of the same opcode. Some specific opcodes reside only in the sequence compiled by certain versions (e.g., Solidity compilers earlier than version 0.4 do not produce opcode `revert`). In addition, the opcode combinations and their positions are different across different Solidity compiler versions. These observations imply that direct birthmarks using signatures or fingerprints on bytecodes may suffer from the susceptibility to the interference introduced by opcodes and weak generalizability, and higher-level yet difference-tolerant representations are needed to address this limitation.

Our goal here is to construct the similarity detection model over Ethereum smart contract bytecodes: we leverage deep neural networks of superior feature learning ability to learn the latent representation from the abstracted opcode sequence for each smart contract's bytecodes, and devise SNN to supervise representation learning and similarity measurement. More specifically, given two smart contracts' bytecodes c_i and c_j, the opcode sequences s_i and s_j are first extracted and abstracted from c_i and c_j respectively, and then jointly embedded into unified vector space so that we can reasonably measure the similarity between them. Formally, the similarity measurement of s_i and s_j can be formulated as:

$$s_i \xrightarrow{\phi(s_i)} \mathbf{x}_i \rightarrow \mu(\mathbf{x}_i, \mathbf{x}_j) \leftarrow \mathbf{x}_j \xleftarrow{\phi(s_j)} s_j \tag{1}$$

where $\phi : s \rightarrow \mathbf{x} \in \mathbb{R}^d$ is representation learning function to map opcode sequence s into d-dimensional vector space, and $\mu(\cdot)$ is similarity measuring function. We exploit SNN to design $\mu(\cdot)$. The similarity detection can thus be stated in the form of $\mu : (\phi(s_i), \phi(s_j)) \rightarrow y$, which outputs the similarity score of the input sample pair in the class space $y \in \{0, 1\}$ representing the different and similar classes respectively. That is, if s_i and s_j are similar, then $\mu(\phi(s_i), \phi(s_j)) \rightarrow 1$; otherwise, $\mu(\phi(s_i), \phi(s_j)) \rightarrow 0$.

III. Methodology

In this section, we technically detail how we perform representation learning and similarity detection on smart contract bytecodes though SNN in our designed model SmartSD. Fig. 1 specifies the overview of the SmartSD.

A. Data Preprocessing

In order to prepare the smart contracts for our experimental evaluations on SmartSD, we first collect our dataset using Etherscan, which is a block explorer and analytics platform for Ethereum. More specifically, the HTML pages describing the transaction information of smart contracts are first crawled and parsed, such that the data regarding smart contracts' bytecodes, and their source codes and compiler versions used are then collected. Each contract's bytecodes are encoded by a string of hexadecimal bytes, which do not reflect the underlying operations of Ethereum. Therefore, it is inadvisable to analyze these binaries directly on their raw bytes. In this regard, we disassemble the contract's bytecode into assembly EVM instructions first, and perform the representation learning on them. Currently, the EVM defines about 142 instructions, and there are more than 100 instructions to be extended. For the ground truth preparation, we compile a smart contract source code to generate bytecodes using compilers of different versions, where we consider bytecode pairs from them as positive (similar) data and bytecodes compiled from different source code as negative (different) data. More details are presented in Section IV-A.

B. Representation Learning

Given the instruction set extracted from the bytecode of each smart contract, it is reasonable to learn the embedding from the sequence as the desired representation for each smart contract. However, such an implementation applied directly on the

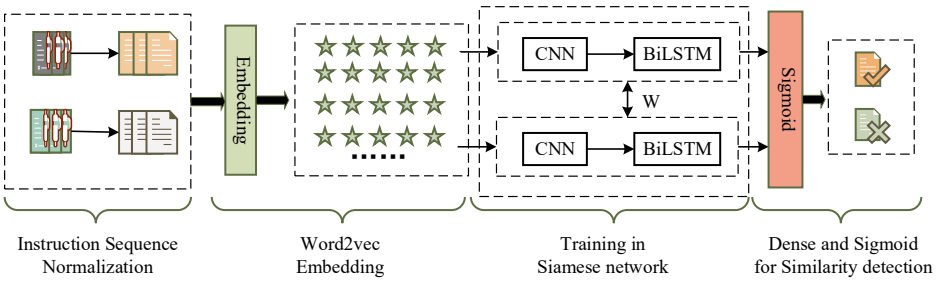


Fig. 1. The overview of SmartSD

instruction set can not expressively capture the correlations and variations among smart contracts. As we want to make features reflect the smart contract semantics instead of functionality, using all instructions as they exactly appear may expose us to the exhaustive functionality information. This immensely increases the representation learning complexity, and decreases the embedding expressiveness, which may degrade the successive SNN performance in turn. To this end, the instruction set of each smart contract is initially abstracted to the sequence of opcodes. Afterwards, the opcode sequence is fed to deep learning framework for representation learning.

1) Instruction Abstraction: Each instruction contains an opcode followed by some low-frequency tokens (e.g., numerical constants, memory addresses, and special strings), which can be seen on the left-hand side of Fig. 2. These low-frequency tokens are random and have a very insignificant relationship with the semantics of the program. In this respect, we design the rule to abstract each instruction for the subsequent representation learning as follows: the opcode remains unchanged, and all the non-opcode tokens in the instruction are removed. An example in Fig. 2 illustrates the instruction abstraction processing: the instruction sequence on the left is disassembled from smart contract bytecodes; the abstracted instruction sequence on the right is simply composed of opcodes. For example, using the designed abstraction rule, the instruction "PUSH1 0x80" becomes "PUSH1", and the instruction "PUSH1 0xf" is changed to "PUSH1". After the instruction abstraction, we can represent a smart contract c as an opcode sequence $s = \{op_1, op_2, op_3, \cdots, op_n\}$. In the sequence, each op_i represents the opcode of the i_{th} instruction within n total number of instructions.

2) Opcode Embedding: After getting the opcode sequences for smart contracts, we further transform them into numerical embedding space that neural network is able to understand and process. To this end, we first perform embedding operation to map each unique opcode to a vector, such that the opcode sequence can be comprehensively represented. Specifically, SmartSD employs the representative skip-gram model [17] to learn opcode representations to encode their contextual relatedness. Given a set of opcode sequences, each of which is $s = \{op_1, op_2, op_3, \cdots, op_n\}$, we feed them to the skip-gram model, and obtain a k-dimensional vector for each unique opcode by evaluating an opcode's neighborhood co-occurrence

PUSH1 0x80	PUSH1
CALLVALUE	CALLVALUE
DUP1	DUP1
ISZERO	ISZERO
PUSH1 0xf	PUSH1
.....	.....
DUP1	DUP1
REVERT	REVERT
PUSH6 0x627a7a723058	PUSH6 0x627a7a723058
SHA3	SHA3
PUSH27	PUSH27
0x227f8a994da323974f6	0x227f8a994da323974f6
ba452a809c3e4b68349ee	ba452a809c3e4b68349ee
3fdd985e5a911e	3fdd985e5a911e

Fig. 2. Abstraction for smart contract instructions.

within a window w conditioned on its current embedding. In this way, the learning objective of skip-gram is defined as:

$$\underset{\psi}{\operatorname{argmin}} - \sum_{-w \le j \le w, i \ne j} \log p(op_{i+j} | \psi(op_i)), \qquad (2)$$

where $\psi(op_i)$ is op_i's current embedding. After opcode embedding, the opcode sequence of each smart contract can be converted to an $n \times k$-dimensional vector.

3) Representation Learning for Opcode Sequences: Among neural networks, convolutional neural network (CNN) [16] can capture the local correlation, while Long Short-term Memory (LSTM) [9] can encode the sequential dependency, which best fit in our problem. We thus design a model that leverages the advantages of CNN and LSTM over the opcode embedding sequences for smart contract representation learning. The model network structure diagram for representation learning is shown in Fig. 3.

We first enable CNN, which stacks a convolutional layer and a normalization layer, to refine the opcode embedding sequence of each smart contract with locally aggregated opcode information. In this way, it crafts more informative and higher-level embedding space to facilitate the following sequence modeling. To characterize the interactions among different opcode grams, we further integrate filters of different kernel sizes into CNN to enrich the feature semantics for smart contracts. Taking the opcode sequence embedding matrix $\mathbf{S} \in \mathbb{R}^{n \times k}$, the convolutional layer adopts m filters of shape $l \times k$ to perform convolution operations on $\mathbf{S}$, and formulates a new embedding matrix $\mathbf{S}^* \in \mathbb{R}^{(n-l+1) \times m}$ with kernel size l.

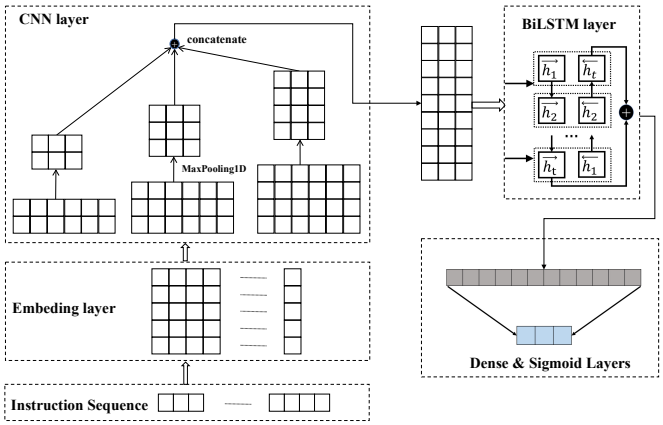


Fig. 3. Representation learning using CNN and BiLSTM

To extract multi-view feature patterns from 2, 3 and 4 opcode grams, we employ kernels of size 2, 3 and 4 to convolute $\mathbf{S}$.

Afterward, the resulting feature matrix $\mathbf{S}^* = [\mathbf{e}_1, \cdots, \mathbf{e}_n]^T$ from CNN is fed to bidirectional LSTM (BiLSTM) to embed the sequential dependency, and output the desired representation. The BiLSTM proceeds by (1) reading $\mathbf{S}^*$ through the composite non-linear transformations $\mathcal{H}$ to learn a hidden vector $\mathbf{h}_t$ at timestep t: $\mathbf{h}_t = \mathcal{H}(\mathbf{e}_t, \mathbf{h}_{t-1}), \quad \mathbf{h}_t \in \mathbb{R}^d$ [9]; (2) devising two LSTMs with one processing the sequence in a forward direction and the other in a backward direction to jointly capture bidirectional dependencies and provide additional context to the network; and (3) concatenating the forward and backward hidden vectors at timestep t into new $\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}\right]$. As it entirely reads the input sequence in both directions, the hidden states $\mathbf{h}_n$ at the last timestep act as the summary vector to represent the opcode sequence.

C. Supervised Learning using Siamese Neural Networks

Under the supervise-learning setting, we use siamese neural networks (SNN) [1], [12] to chain and optimize the full learning procedure, including representation learning, similarity measurement, and similarity prediction. There are four reasons behind this network choice. (1) SNN has been successfully deployed in the similarity-based applications, such as zero-shot and few-shot image recognition. (2) SNN supports back-propagation optimization for the aforementioned representation learning process to update its parameters. (3) SNN of twin networks trains the unilateral network by sharing weights [28]; this ensures that smart contract similarity measurement can process two opcode sequences consistently and the weights of both parties are consistent as well. (4) After obtaining the high-level representations of smart contracts, SNN can calculate their similarity in a parameterized manner [22], instead of simply calculating the similarity scores without fine-tuning in an error-prone way.

In our model design, we elaborate a deep SNN, whose identical subnetworks receive two opcode sequence embedding matrices $\mathbf{S}_i$ and $\mathbf{S}_j$ of smart contract bytecodes c_i and c_j, and pass them through CNN and BiLSTM in succession to learn the representations $\mathbf{x}_{s_i}$ and $\mathbf{x}_{s_j}$ respectively that extract high-level and difference-tolerant features. Different from the conventional SNN performing direct similarity metric computes [4], our network's top conjoining layer devises a multilayer perceptron (MLP) [8] stacking multiple fully connected layers to fuse features as:

$$\mathbf{x} = \left[\mathbf{x}_{s_i}; \mathbf{x}_{s_j}\right] \tag{3}$$

and measure the similarity using:

$$\mu_\theta\left(\mathbf{x}_{s_i}, \mathbf{x}_{s_j}\right) = \mathrm{MLP}_\theta\left(\mathbf{x}\right) \tag{4}$$

where θ are parameters introduced by the network. This is followed by a sigmoid activation function mapping onto the interval $[0, 1]$, which is used to minimize cross-entropy loss. Given the training opcode sequence pair from the corresponding smart contracts' bytecodes $(s_i, s_j) \in D$, and the similarity label $y \in Y$ where $y = 0$ means that s_i and s_j are different, and $y = 1$ means that s_i and s_j are similar, the cross entropy loss of our similarity detection (i.e., binary classification problem) can be defined as:

$$\mathcal{L} = -\sum_{(\mathbf{X}_p^I, \mathbf{X}_q^I) \in \mathcal{D}} y \log(P) + (1 - y) \log(1 - P) \tag{5}$$

$$\text{s.t.} \quad P = \sigma(\mu_\theta(\mathbf{x}_{s_i}, \mathbf{x}_{s_j})) \tag{6}$$

where σ is the sigmoid activation function, and the parameters introduced by representation learning and similarity measurement can be comprehensively updated via gradient descent algorithms (e.g., Adam).

From the model formulation, it is worth remarking two significant advantages yielded by our methodology: (1) smart contract representation learning and similarity measurement between sample pairs are automated and advanced by deep learning frameworks without prior domain knowledge; and (2) the task-specific representations learned by the designed networks can tolerate the difference caused by compilers and their optimization options, and generalize well to unseen data..

IV. Evaluation

A. Dataset Preparation and Experimental Settings

As a supervised deep learning based scheme is adopted by SmartSD, a large dataset consisting of totally 72,612 samples is thus constructed to boost the training and testing of our method. Specifically, the following steps are enforced in preparing the samples with ground-truth labels:

- To correctly prepare positive (contract pairs that are really similar) and negative (contract pairs that are indeed different) samples, we utilize smartEmbed [6], a tool that detects smart contract clones based on their Solidity source code, to identify similar and dissimilar smart contracts crawled from Etherscan. In our setting, their detected smart contract pairs, which are not identical but with similarity values greater than 0.95, are taken as candidates of positive samples CP; while the detected pairs with similarity values smaller than 0.35, are taken as candidates of negative samples CN.

- On the basis of the candidate positive and negative samples, we further retrieve their corresponding runtime bytecode from the public Ethereum Cryptocurrency database[2] that is hosted on the Google BigQuery by feeding in their contract addresses. The bytecodes of a candidate smart contract pair is then organized into a triplet of $\langle bin(p), bin(q), l\rangle$, where $l \in \{-1, +1\}$ is the ground truth label that indicates whether the smart contracts p and q belongs to the positive or the negative pair, while $bin(.)$ denotes the runtime bytecode of a smart contract.
- To improve the ability of the trained model in dealing with the adverse impacts from different compiler settings, we further attempt to incorporate positive and negative samples by compiling a smart contract's source code with varying compiler settings. To this end, we randomly pick equivalent number of smart contract pairs from CP and CN, and then try to compile them with varying Solidity compiler versions as well as setting the *–optimize* option on and off. Specifically, 5 different Solidity compiler versions, including *0.4.24*, *0.5.6*, *0.6.4*, *0.7.2* and *0.8.2*, are set up to compile the source code of each picked smart contract individually[3]. The successfully compiled ones are then combined to make up positive and negative triplets accordingly.

With these above steps, we finally manage to produce 42,082 pairs of positive samples and 30,530 pairs of negative samples, as our dataset.

For experimental settings, the dataset is randomly divided into training, validation and test set at a ratio of 80%, 10%, and 10%. The model is trained using a RTX3090 GPU with the Adam optimizer. The batch size is set to 64, and the initial learning rate is set to 0.001. In each epoch, we shuffle the training samples while calculate the accuracy on the validation set. Besides, to avoid the over-fitting and non-convergence problems, early stopping is enforced that stops model training right after the epoch that the validation accuracy no longer improves. Finally, the model with the highest accuracy witnessed during all the epochs is adopted, with which frequently used performance metrics including accuracy, precision, recall, and F1 score are evaluated and reported on the test set. Also, the opcode embedding model is trained using 100 epochs and 6 context window size.

B. Evaluation Results

In this section, we report the performance of SmartSD. In addition, we compare the performance of SmartSD with variant models by simply substituting the CNN+BiLSTM structure in our SmartSD with other widely used deep neural network structures, including the pure CNN, BiLSTM, GRU and Bi-GRU. We denote them as SmartSD$_{CNN}$, SmartSD$_{BiLSTM}$, SmartSD$_{GRU}$ and SmartSD$_{BiGRU}$, respectively.

TABLE I
PERFORMANCE COMPARISON WITH VARYING NEURAL NETWORK STRUCTURES IN SMARTSD

Model	*Accuracy*	*Precision*	*Recall*	*F1-score*
SmartSD	**98.37%**	**0.9889**	**0.9813**	**0.9850**
SmartSD$_{CNN}$	93.09%	0.9321	0.9119	0.9219
SmartSD$_{BiLSTM}$	95.20%	0.9536	0.9383	0.9459
SmartSD$_{GRU}$	93.14%	0.9352	0.9071	0.9209
SmartSD$_{BiGRU}$	94.32%	0.9479	0.9337	0.9407

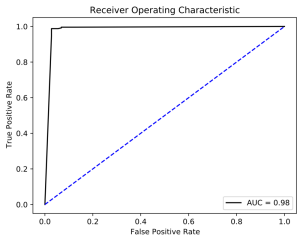


Fig. 4. Performance regarding ROC

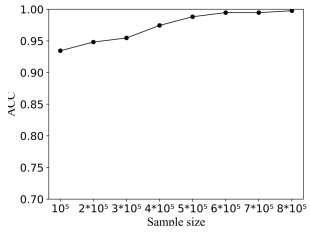


Fig. 5. Training size analysis

Table I summaries the experimental results. As the data show, SmartSD as well as its substituted models all exhibit rather good detection performance with respect to the evaluation metrics mentioned in Section IV-A. Their accuracy values generally compete with or outperform the 93.27% accuracy value as reported by EClone [14], [15], a smart contract similarity detection method that adopts inefficient and sophisticated symbolic execution techniques. This indicates the potency of adopting a deep learning based way to achieve smart contract similarity detection task. Especially, the original SmartSD that adopts a CNN+LSTM structure for encoding the opcode sequence of a smart contract's bytecodes, outperforms all the other substitute models with a relatively obvious margins. Its 98.37% high detection accuracy and 0.9850 F1-score value indicate the superiority of combing CNN and LSTM structure to capture compiler-setting-agnostic features, which makes our method highly resilient against the disturbance of varying compiler settings. Additionally, as depicted in Fig. 4 for the ROC curve, SmartSD achieves a very high AUC (Area Under the Curve) value of 0.98.

The impact of the training set size on the detection performance of SmartSD is also evaluated, by training SmartSD with gradually increased number of training samples and observing corresponding detection accuracy on the same test set. As depicted in Figure 5, as the number of training samples increases, the detection performance of SmartSD increases. It indicates the importance of abundant data for deep learning based methods, while the availability of the massive diversiform open-sourced smart contracts on Etherscan makes deep learning especially suitable for our problem.

[2]https://cloud.google.com/blog/products/data-analytics/ethereum-bigquery-public-dataset-smart-contract-analytics

[3]As pointed out in a recent study [21], there are many minor Solidity compiler versions within each major version (For example, besides 0.4.24, there are 26 more officially released minor versions in the major version 0.4.x), but their impacts on the compiled bytecodes are insignificant, our used minor versions are thus randomly selected as long as there is one for each major version. Also, note that not all compiler versions can always successfully compile each smart contract, especially earlier versions are not used considering their immaturity and the high failure rate in compiling the smart contract in our dataset.

V. Related Work

Similarity analysis of smart contracts: Code similarity analysis has always been a long-term research topic, but there have been few studies [6], [14], [15] conducted on the emerging smart contracts. Liu et al. [14], [15] defined the concept of smart contract birthmarks by borrowing the typical definition of software birthmark. They proposed a symbolic transaction sketch technique to achieve smart contract similarity detection and DApp (Decentralized Application) clone detection on the bytecode level. Different from their methods that resorts to symbolic execution and expert domain knowledge, SmartSD recurs to the deep neural networks' powerful learning ability and the availability of many smart contract to achieve high accurate and efficient similarity detection while with less human intervention. Gao et al. [6] proposed to use unsupervised embedding algorithms including word2vec and Fasttext to encode smart contracts into numerical vectors, on the basis of which similarity between smart contract pairs can be efficiently computed with an Edulidean distance based similarity metric. Different from their works that can only operates on the Solidity source code, we adopt a deep siamese neural network architecture that works on the actually deployed smart contracts' bytecodes, with the aim of defeating the disturbance from varying compiler settings.

Smart contract bug detection: With the developments of smart contracts and the widely used yet maturing programming languages, their defects and security issues have attracted major research attentions for the solutions. Apart from these conventional methods [13], [18], [23], [24] that generally detect bugs or vulnerabilities based on symbolic execution, formal verification, and manually constructed bug patterns or specifications, Gao et al. [7] attempts to detect/retrieve bugs from smart contracts' source code by checking the similarity of the smart contract against known buggy contracts. SmartSD can accurately detect the similarity of smart contracts directly on the deployed bytecode, thus it is promising to be applied to achieve similarity checking based known bug search in the scenario that smart contracts' source codes are unavailable.

VI. Conclusion

In this paper, we propose to detect the similarity of Ethereum smart contracts and build up a bytecode-level model SmartSD using deep siamese neural network to supervise the representation learning and the similarity measurement process. The experimental results show that SmartSD achieves 98.37% high detection accuracy and 0.9850 F1-score, which demonstrate its effectiveness of smart contract similarity detection; SmartSD also significantly outperforms the baseline models, and is computationally tractable and effectively mitigates the interference caused by compilers.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61702414), the Science and Technology of Xi'an (2019218114GXRC017CG018-GXYD17.16), the Natural Science Basic Research Program of Shaanxi (2022JM-342, 2018JQ6078, 2020JM-582), the International Science and Technology Cooperation Program of Shaanxi (2019KW-008), the Key Research and Development Program of Shaanxi (2019ZDLGY07-08), and Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

References

[1] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *IJPRAI*, vol. 7, no. 04, pp. 669–688, 1993.
[2] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, "Defining smart contract defects on ethereum," *TSE*, 2020.
[3] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *SANER*. IEEE, 2017, pp. 442–446.
[4] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.
[5] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in *ICSE*, 2020, pp. 530–541.
[6] Z. Gao, V. Jayasundara, L. Jiang, X. Xia, D. Lo, and J. Grundy, "Smartembed: A tool for clone and bug detection in smart contracts through structural code embedding," in *ICSME*, 2019.
[7] Z. Gao, L. Jiang, X. Xia, D. Lo, and J. Grundy, "Checking smart contracts with structural code embedding," *TSE*, pp. 1–18, 2020.
[8] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017.
[9] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
[10] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the ethereum smart contract ecosystem," in *FC*, 2020.
[11] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts." in *Ndss*, 2018, pp. 1–12.
[12] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," vol. 2. Lille, 2015.
[13] J. Krupp and C. Rossow, "teether: Gnawing at ethereum to automatically exploit smart contracts," in *USENIX Security*, 2018, pp. 1317–1333.
[14] H. Liu, Z. Yang, Y. Jiang, W. Zhao, and J. Sun, "Enabling clone detection for ethereum via smart contract birthmarks," in *ICPC*, 2019.
[15] H. Liu, Z. Yang, C. Liu, Y. Jiang, W. Zhao, and J. Sun, "Eclone: Detect semantic clones in ethereum via symbolic transaction sketch," in *ESEC/FSE*, 2018.
[16] Y. Luan and S. Lin, "Research on text classification based on cnn and lstm," in *ICAICA*. IEEE, 2019, pp. 352–355.
[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.
[18] M. Mossberg, F. Manzano, E. Hennenfent, A. Groce, G. Grieco, J. Feist, T. Brunson, and A. Dinaburg, "Manticore: a user-friendly symbolic execution framework for binaries and smart contracts," in *ASE*, 2019.
[19] K. Sliwak, "Smart contract reentrancy: Thedao," https://medium.com/@zhongqiangc/smart-contract-reentrancy-thedao-f2da1d25, 2021.
[20] Solidity, "Github - ethereum/solidity: Solidity, the contract-oriented programming language," https://github.com/ethereum/solidity, 2021.
[21] Z. Tian, J. Tian, Z. Wang, Y. Chen, H. Xia, and L. Chen, "Landscape estimation of solidity version usage on ethereum via version identification," *IJIS*, 2021.
[22] Z. Tian, Q. Wang, C. Gao, L. Chen, and D. Wu, "Plagiarism detection of multi-threaded programs via siamese neural networks," *IEEE Access*, vol. 8, pp. 160 802–160 814, 2020.
[23] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smartcheck: static analysis of ethereum smart contracts," in *WETSEB*, 2018, pp. 9–16.
[24] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, and et al., "Securify: practical security analysis of smart contracts," in *CCS*, 2018.
[25] M. Wohrer and U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity," in *IWBOSE*, 2018, pp. 2–8.
[26] G. WOOD, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, pp. 1–32, 2014.
[27] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, pp. 1–32, 2014.
[28] L. Zhang, Z. Feng, W. Ren, and H. Luo, "Siamese-based bilstm network for scratch source code similarity measuring," in *IWCMC*. IEEE, 2020, pp. 1800–1805.

An Information Flow Security Logic for Permission-Based Declassification Strategy

Zhenheng Dong[1], Yongxin Zhao[1*] and Qiang Wang[2]
[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
[2] Chinese Academy of Military Science, Beijing, China

***Abstract*—With the increasing popularity of smartphones and the rapid development of mobile network, ensuring the security of mobile applications becomes more and more important, which has received substantial attention from both academia and industry. Information flow security, as a prominent approach to system and network security, aims at ensuring high security level information would not be accessed by analyzing the information with lower security levels. In this paper, we design a novel information flow security logic to reason about the security of mobile applications, leveraging on the idea of permission based declassification. Firstly, we propose a formal language with permission check branches, through which the access to the confidential information can be controlled. Then we present our novel information flow security logic based on the permission based declassification strategy, which can make the reasoning more precise by degrading the security level of the specific information. Finally, we demonstrate the usability of our logic via examples.**

***Index Terms*—Information Flow Security, Formal Language and Logic, Permission-Based Declassification, Mobile Applications**

I. Introduction

Information flow security is a prominent approach to system and network security. Given the fact that system components can be classified into different security levels, among which confidential information (e.g., private data resource) is marked as high security level, whereas public information (e.g., public program variable) is marked as low security level, the basic idea of information flow security is to ensure confidential information with high security level can not be obtained by analyzing the observable information with lower security levels. In the past decades, information flow security has been investigated and applied to various fields, e.g., operating systems [1], Web services [2] and cloud computing [3].

Among all the approaches to guarantee the information flow security, formal methods have been widely used and can be roughly divided into three categories, i.e., process algebra based methods, type system based methods and semantic based methods [4]. The work in [5] described four approaches to verifying security protocols based on process algebra. In [6] the authors used type system to check semantic errors in programs and to catch errors in code before program execution. In [7] the authors implemented a secure type system implementing integrity information flow control. The basic idea of semantic based methods is to bind a security level to program variables, and to adjust the security level according to the semantics of program operations dynamically. If there is no information leakage from the high security level to the low security level, the program access to the system is considered to be secure. The work in [8] proposed a secure semantic web service model by analyzing the web service security. In [9] the COVERN logic was proposed based on the lock mechanism for the security analysis of shared memory in concurrent programs. In [10] a non-blocking algorithm was proposed to avoid the use of locks on shared variables and data structures. Inspired by these previous work, we adopt the semantic based method to solve the information flow security in mobile applications.

In this paper, we propose a semantic-based approach to the information flow security among sequential mobile applications. Particularly, inspired by the literature [11] and [12], our approach introduces a permission-based declassification strategy to ensure that information will only be legally accessed by applications who have been granted the permissions. Throught the approach, the information can be downgraded from the origianl high security level to make the logical reasoning more accurate. The contributions of this work are as follows:

- **Permission-based formal language for information flow security:** The language abstracts application programs into functions, and uses function calls to represent information interactions among applications. Permissions are the basis for whether messages can be obtained in information interaction. In addition, the operational semantics of the language are strictly formulated, which is helpful for the formulation of reasoning logic.
- **Logic rules supporting declassification strategy:** This is the first semantic-based permission-dependent information flow security method, which adds the content of permission into the traditional information flow logic. In addition, the logic also adds a declassification strategy to make the reasoning more precise.

The rest of this paper is organised as follows. In Section II, we provide some running examples to illustrate the intuitive idea of our approach. A permission-based formal language is presented in Section III and its semantics is given in Section IV. In Section V, we discuss our novel informal flow security logic. Section VI illustrates the usability of our logic via examples and Section VII concludes the paper.

*Corresponding author: yxzhao@sei.ecnu.edu.cn
DOI reference number: 10.18293/SEKE2022-134

II. Running Examples

Before introducing the language and logic, we informally describe its core ideas and practical problems that can be solved through some examples.

The current popular definition of information flow security is non-interference [13], that is, if only the input changes, the attacker cannot observe the difference in execution results [14]. However, for mobile applications, traditional information flow security is not applicable due to the need for nested calls of programs, as shown in the example in Table I.

TABLE I: An example of traditional information flows

```
String getInformation(String name){
        String information;
        if(verifyPass(READ_PASSWORD))
                information=name.information;
        else
                information="";
        return information;
}
```

We can think of this example as a login service for social APPs, such as WeChat, instagram, etc., where the parameter $name$ is the username, and the function $verifyPass()$ is for authorization verification. If the password verification is passed, then the information $name.information$ of the user is returned. Otherwise, an empty string is returned. It is obvious that the user information is confidential and its security level should be high, while the security level of an empty string should be low. Because the traditional information flow security requires the two branches of the conditional statement to have the same security level, the non-interference can be guaranteed. At this time, if the information level is high, then the information can be used by other applications as the return value, and there will be a great security risk. Conversely, if the information level is low, either the application cannot obtain useful information, or there is an information leakage problem of assigning a high security level to a variable with a low security level. Therefore, a new strategy is needed to solve the resource acquisition problem of the application.

To solve this problem, the work in [15] proposed a type system that incorporates permissions in function types. And another work in [16] also proposed a type system that solves the problem of typing non-monotonic policies without resorting to downgrading or declassifying the information. Inspired by the above research, we define a statement that supports declassification assignment, as shown in Table II.

TABLE II: An example of our method

```
A.funA(name){
        init information=0;
        in{
                check(p){
                        then  information:=↓ name.information;
                        else  information:=0;
                }
                return information;
        }
}
```

We introduce a permission access control mechanism, and use the Check statement to replace the If-condition in the traditional information flow. If the application contains the permission p, then we can lower the security level by declassifying the assignment statement of the high security level $name.information$, and assignthe declassified variable to information. This can ensure that the return value of each function is a low security level, which ensures the security of the information flow of the system, and also solves the problem of application resource acquisition.

III. The Formal language

In this section, we present a formal language for a permission-based approach to information flow security.

The syntax of expressions is given below:

$$e ::= v \mid x \mid e \ op \ e \mid \overline{e}$$

Where v represents an integer value, x represents a variable, op is a binary operator defined on two expressions, $\overline{e} = [e_1, e_2, ..., e_n]$ represents an expression tuple, and each element in the tuple is an expression respectively.

The syntax of the command statement is given in the following:

$$\begin{aligned} c ::= \quad & skip \mid x := e \mid c_1; c_2 \mid if \ e \ then \ c_1 \ else \ c_2 \\ & \mid while \ e \ do \ c \mid init \ x = e \ in \ c \mid x := call \ A.f(e) \\ & \mid x :=\downarrow e \mid check(p) \ then \ c_1 \ else \ c_2 \end{aligned}$$

Where $skip$ means the empty statement does nothing. $init\ x := e\ in\ c$ is the definition statement of a local variable x, and the program block c represents the scope of the local variable x. $x := call\ A.f(e)$ means to call the function $f(e)$ of the application A, the expression e is the incoming parameter of the function, and uses the variable x receives the return value of the function call. $x :=\downarrow e$ represents the declassification assignment statement, which means that the expression e with high security level is allowed to downgrade the security level and assign it to the variable x. $check(p)\ then\ c_1\ else\ c_2$ represents the permission check statement, check whether the atomic permission p is included in the permission context, if it does, execute the statement c_1, otherwise execute c_2. Also, other statements are not much different from those of other programming languages.

A function definition is in the following:

$$F ::= A.f(\overline{input})\{init\ output = 0\ in\ \{c\ ;\ return\ output\}\}$$

Where $A.f$ denotes the function of application A whose function name is f. $\overline{input}$ is the formal parameter of the function, c is the execution statement of the function body, $output$ is the local variable and the return value of the function, and $\{c; return\ output\}$ is its scope. We only consider the closed function in this language, that is the variables that appear in c will only be variables introduced into the parameter input or local variables within the function.

IV. Operational semantics

A. Semantic Model

In the semantic model, we define a system state μ as a tuple $\langle mem, tr\rangle$, where mem represents memory and tr represents event trace.

The memory mem_μ denotes a set of assignments for all variables in the state μ, that is $mem_\mu = [x_1 \mapsto v_1, x_2 \mapsto v_2, \ldots, x_n \mapsto v_n]$, where $x_1, x_2, \ldots, x_n$ represents a finite set of all variables in the system, and $mem_\mu(x_1)$ represents the value of the variable x_1 under the state of the system μ.

We use a sequence tr_μ to save all the assignment events from the initial state to the current state of the system. In this study, the event $event$ consists of two kinds of events: First, the ordinary assignment event $ASG\langle lvl, x, e\rangle$, which means to assign the value of the expression e to the variable x whose security level lvl. Second, the classification assignment event $DCF\langle lvl, x, e\rangle$ means that the value of the expression e is declassified and assigned to the variable x, and after declassification, the security level of variable x is lvl. The sequence of each historical event is called a trace, which is a finite collection of events.

B. Operational Semantics

We refer to [17] and [18] for our semantics. Our semantics is divided into two levels: expressions and statements.

The evaluation of the expressions are given in Table III. The semantic judgment of the expression has the form of $\mu \vdash e \Rightarrow v$, where μ represents the system state, e is the expression, and v is the value of the expression, which can be interpreted as in the state μ, the value of the expression e is v.

TABLE III: Evaluation of Expressions

$$\textbf{(Value)}\ \frac{}{\mu \vdash v \Rightarrow v}$$

$$\textbf{(Variable)}\ \frac{}{\mu \vdash x \Rightarrow mem_\mu(x)}$$

$$\textbf{(Tuple)}\ \frac{\mu \vdash e_i \Rightarrow v_i\ (1 \le i \le n)}{\mu \vdash \overline{e} \Rightarrow \overline{v}}$$

$$\textbf{(BiOp)}\ \frac{\mu \vdash e_1 \Rightarrow v_1 \quad \mu \vdash e_2 \Rightarrow v_2 \quad v_1\ op\ v_2 = v_3}{\mu \vdash e_1\ op\ e_2 \Rightarrow v_3}$$

(Value) indicates that in the system state μ, the value of the numerical expression v is v.

(Variable) indicates that in the system state μ, the value of the variable x is the value of x in the memory under the system state, which can be expressed as $mem_\mu(x_1)$.

(Tuple) indicates that in the system state μ, when $1 \le i \le n$, the value of the expression e_i is v_i respectively, then the value of the tuple expression $\overline{e}$ is expressed as $\overline{v}$.

(BiOp) indicates that in the system state μ, if the value of e_1 is v_1, the value of e_2 is v_2, and the value of v_1 and v_2 calculated by the binary operator is v_3. Then in this state, the result of the binary operation between $e1$ and $e2$ is $v3$.

The operational semantics of command statements are given in Table IV. The operational semantic judgment of the command statement has the form of $A, P \vdash (\langle mem, tr\rangle, c) \rightarrow \langle mem', tr'\rangle$. Where A represents the application, and the permission set P denotes the permission context, that is the set of permissions of the application which invokes the function of A. $\langle mem, tr\rangle$ represents a system state μ, tr represents the event execution sequence. This judgment means that in the environment of the application A with the permission context P, the system state before executing the statement c is $\langle mem, tr\rangle$, and after executing, it becomes $\langle mem', tr'\rangle$.

(Skip) and **(Seq)** are relatively simple, and they are not very different from the operational semantics of general sequential languages, so we will not go into details here. **(If)** and **(While)** use the expression e to determine whether the condition is true. For simplicity, we use TRUE and FALSE as the distinguishing criterion.

Two assignment statements correspond to two kinds of events in the system, that are ASG and DCF. When executing a normal assignment statement **(AssignN)**, we should add ASG event after the current trace tr. However, when executing the declassification assignment statement **(AssignD)**, we add DCF event after tr. Depending on the trace of the event history, we can distinguish which type of assignment statement is currently executing.

The local variable definition statement **(DefL)** means to define a local variable x whose initial value is the value v of the expression e. The scope of the variable x is only within the statement block c. Its operational semantics are to assign the value v of the expression e to the variable x, and execute the statement c under this premise. After execution ends, local variables should be removed from the memory. $mem' - x$ means to delete the local variable x from memory mem'.

The permission check statement **(Check)** is similar to an **(If)** statement. Where p is an atomic permission in the system permission set. Which branch of the statement is executed depends on the relationship of the atomic permission p to the permission context P. If P contains permission p, that is, $p \in P$, execute the statement c_1, otherwise execute c_2.

(Call) is more complex than the operational semantics of other statements. When application A calls function f of application B, first we need to find function f in all functions of application B. Then we use P_A to present the permission set of the application A, after that we give permission set P_A to application B, and execute function body in application B. Finally, assign the return value to the variable x. In short, application A calls the function of application B, we should check whether the corresponding permission exists in application A during the process of executing the function of B. This means that the permission set of application cannot be passed in recursive calls to functions.

V. The Logic

A. Access Control Model

In traditional information flow security, we introduce permissions for access control. We use $\mathcal{P}$ to denote a finite set of all permissions in the entire system, that is, $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. P represents the permission context, and $P \subseteq \mathcal{P}$. We require

TABLE IV: Operational Semantics of Command Statements

$$\textbf{(Skip)}\ \frac{}{A,P\vdash(\langle mem,tr\rangle,skip)\rightarrow\langle mem,tr\rangle} \qquad \textbf{(AssignN)}\ \frac{\mu\vdash\ e\ \Rightarrow\ v \quad tr'=tr\cdot ASG}{A,P\vdash(\langle mem,tr\rangle,x:=e)\rightarrow\langle mem[x\mapsto v],tr'\rangle}$$

$$\textbf{(Seq)}\ \frac{A,P\vdash(\langle mem,tr\rangle,c_1)\rightarrow\langle mem',tr'\rangle \quad A,P\vdash(\langle mem',tr'\rangle,c_2)\rightarrow\langle mem'',tr''\rangle}{A,P\vdash(\langle mem,tr\rangle,c_1;c_2)\rightarrow\langle mem'',tr''\rangle}$$

$$\textbf{(IfF)}\ \frac{\mu\vdash e\Rightarrow FALSE \quad A,P\vdash(\langle mem,tr\rangle,c_2)\rightarrow\langle mem',tr'\rangle}{A,P\vdash(\langle mem,tr\rangle,if\ e\ then\ c_1\ else\ c_2)\rightarrow\langle mem',tr'\rangle} \qquad \textbf{(IfT)}\ \frac{\mu\vdash e\Rightarrow TRUE \quad A,P\vdash(\langle mem,tr\rangle,c_1)\rightarrow\langle mem',tr'\rangle}{A,P\vdash(\langle mem,tr\rangle,if\ e\ then\ c_1\ else\ c_2)\rightarrow\langle mem',tr'\rangle}$$

$$\textbf{(WhileF)}\ \frac{\mu\vdash e\Rightarrow FALSE}{A,P\vdash(\langle mem,tr\rangle,while\ e\ do\ c)\rightarrow\langle mem,tr\rangle}$$

$$\textbf{(WhileT)}\ \frac{\mu\ \vdash\ e\ \Rightarrow TRUE \quad A,P\vdash(\langle mem,tr\rangle,c)\rightarrow\langle mem',tr'\rangle \quad A,P\vdash(\langle mem',tr'\rangle,while\ e\ do\ c)\rightarrow\langle mem'',tr''\rangle}{A,P\vdash(\langle mem,tr\rangle,while\ e\ do\ c)\rightarrow\langle mem'',tr''\rangle}$$

$$\textbf{(DefL)}\ \frac{\mu\ \vdash\ e\ \Rightarrow v \quad A,P\vdash(\langle mem[x\mapsto v],tr\rangle,c)\rightarrow\langle mem',tr'\rangle}{A,P\ \vdash\ (\langle mem,tr\rangle,init\ x=e\ in\ c)\rightarrow\langle mem'-x,tr'\rangle} \qquad \textbf{(AssignD)}\ \frac{\mu\vdash\ e\ \Rightarrow\ v \quad tr'=tr\cdot DCF}{A,P\vdash(\langle mem,tr\rangle,x:=\downarrow e)\rightarrow\langle mem[x\mapsto v],tr'\rangle}$$

$$\textbf{(CheckF)}\ \frac{p\notin P \quad A,P\vdash(\langle mem,tr\rangle,c_2)\rightarrow\langle mem',tr'\rangle}{A,P\vdash(\langle mem,tr\rangle,check(p)\ then\ c_1\ else\ c_2)\rightarrow\langle mem',tr'\rangle} \qquad \textbf{(CheckT)}\ \frac{p\in P \quad A,P\vdash(\langle mem,tr\rangle,c_1)\rightarrow\langle mem',tr'\rangle}{A,P\vdash(\langle mem,tr\rangle,check(p)\ then\ c_1\ else\ c_2)\rightarrow\langle mem',tr'\rangle}$$

$$\textbf{(Call)}\ \frac{Find(B.f)=B.f(\overline{input})\{init\ output=0\ in\ \{c\ ;\ return\ \ output\}\} \quad \mu\vdash\overline{e}\Rightarrow\overline{v} \quad B,P_A\vdash(\langle mem[\overline{input}\mapsto\overline{v},output\mapsto 0],tr\rangle,c)\rightarrow\langle mem',tr'\rangle}{A,P\vdash(\langle mem,tr\rangle,x:=call\ B.f(\overline{e}))\rightarrow\langle mem[x\mapsto mem(output)],tr'\rangle}$$

that the permission set of each application is statically allocated in the initial state and cannot be modified dynamically.

We divide variables into two categories, one is the variable that requires permission to access, and the other is the variable that can be accessed without permission. For variables that require permission to access, we use the function $\Gamma(x)$ to represent the permission required to access the variable x. For example, when $\Gamma(x)=p$, it means permission p corresponds to the access to the variable x, then when the permission set P contains the permission p, the currently executing application can get access to the variable x through $check(p)$, otherwise the application cannot access the variable x. For variables that do not require permission access, it is not necessary to read the variables through permission control.

We assume that the application running at this time is A. In the case of no function call, the context permission set P is the permission set P_A of the application A. The application A can access all variables that do not require permission access and the variables corresponding to the permissions possessed by its permission set P_A. When making a function call, if the application A calls the function of the application B, the context permission set P is the permission set P_A, and then we use P to execute application B.

B. Information Flow Model

We define the security level lvl for all expressions on the grid $Low\leq lvl\leq High$. First, we define the security level for the expression e. We stipulate that the numeric type v has no specific security level, and its security level depends on the security level of the variable it is assigned to. For the variable x that has defined the permission function $\Gamma(x)$, we require its security level to be the highest $High$. For some variables, we use the function $L(x)$ to define the security level, and use $L(x)$ to represent the highest security level of the data that the variable x can hold at any time. This means that, in all assignments, the security level of the variable x cannot exceed the upper limit of $L(x)$. For the variable x that does not define $L(x)$, we default its security level to any level. When we assign the value of the expression e to the variable x through $x:=e$, x automatically has the security level of expression e. The current security level of the variable x depends on the security level of the last assigned expression. For the binary operation expression $e:=e_1\ op\ e_2$, the security level of the expression e is the one with the higher security level among the two expressions involved in the operation. Similarly, in the expression tuple $e=\{e_1,e_2,e_3,\ldots,e_n\}$, the security level of the expression e is the highest security level held by all expressions in the tuple. And we use lvl_e to denote the current security level of expression e.

We introduce the attacker role to prove the security of the system, we assume that the attacker is a passive attacker who can only get information by observing the execution of the program. Specifically, we assume that the attacker has an attack level of lvl_a, then for all variables with a security level of $lvl\leq lvl_a$, the attacker can observe them. At the same time, if the assignment operation is declassification assignment, the variable after declassification which security level is $lvl\leq lvl_a$ can also be observed by the attacker.

Our logic has judgements of the form $\mu,A,P,lvl_a\ \vdash\ c$, where μ represents the current system state, A represents the name of the application, P represents the permission context, lvl_a represents the security level of the attacker, and c represents a program statement. This judgment is true if and only if the system state is μ, and we execute application A on permission context P, the program text c will not leak information to the lvl_a level.

C. Declassification Model

In order to make the declassification assignment statement be executed safely, we define a declassification predicate:

$$D(lvl_{src},lvl_{des},\mu,c,P).$$

In the definition of the predicate, lvl_{src} represents the security level of the expression before declassification, lvl_{des}

TABLE V: Rules of the logic

(**R-Skip**) $\dfrac{}{\mu, A, P, lvl_a \vdash skip}$ (**R-UAsgN**) $\dfrac{L(x)\ \ is\ \ undefined}{\mu, A, P, lvl_a \vdash x := e}$

(**R-LAsgN**) $\dfrac{lvl_e \leq L(x)}{\mu, A, P, lvl_a \vdash x := e}$ (**R-Seq**) $\dfrac{\mu, A, P, lvl_a \vdash c_1 \quad A, P \vdash (\mu, c_1) \rightarrow \mu' \quad \mu', A, P, lvl_a \vdash c_2}{\mu, A, P, lvl_a \vdash c_1; c_2}$

(**R-IfF**) $\dfrac{\mu \vdash e \Rightarrow FALSE \quad \mu, A, P, lvl_a \vdash c_2 \quad lvl_e \leq lvl_a}{\mu, A, P, lvl_a \vdash if\ e\ then\ c_1\ else\ c_2}$ (**R-IfT**) $\dfrac{\mu \vdash e \Rightarrow TRUE \quad \mu, A, P, lvl_a \vdash c_1 \quad lvl_e \leq lvl_a}{\mu, A, P, lvl_a \vdash if\ e\ then\ c_1\ else\ c_2}$

(**R-WhileF**) $\dfrac{\mu \vdash e \Rightarrow FALSE \quad lvl_e \leq lvl_a}{\mu, A, P, lvl_a \vdash while\ e\ do\ c}$

(**R-WhileT**) $\dfrac{\mu \vdash e \Rightarrow TRUE \quad \mu, A, P, lvl_a \vdash c\ ;\ while\ e\ do\ c \quad lvl_e \leq lvl_a}{\mu, A, P, lvl_a \vdash while\ e\ do\ c}$

(**R-DefL**) $\dfrac{\mu \vdash e \Rightarrow v \quad \langle mem[x \mapsto v], tr\rangle, A, P, lvl_a \vdash c}{\mu, A, P, lvl_a \vdash init\ x = \ e\ in\ c}$ (**R-AssignD**) $\dfrac{D(lvl_e, L(x), \mu, x :=\downarrow e, P)}{\mu, A, P, lvl_a \vdash x :=\downarrow e}$

(**R-CheckF**) $\dfrac{p \notin P \quad \mu, A, P, lvl_a \vdash c_2}{\mu, A, P, lvl_a \vdash check(p)\ then\ c_1\ else\ c_2}$ (**R-CheckT**) $\dfrac{p \in P \quad \mu, A, P, lvl_a \vdash c_1}{\mu, A, P, lvl_a \vdash check(p)\ then\ c_1\ else\ c_2}$

(**R-Call**) $\dfrac{Find(B.f) = B.f(\overline{input})\{init\ output=0\ in\ \{c\ ;\ return\ \ output\}\} \quad \mu \vdash \overline{e} \Rightarrow \overline{v} \quad \langle mem[\overline{input} \mapsto \overline{v}, output \mapsto 0], tr\rangle, B, P_A, lvl_a \vdash c \quad lvl_{output} \leq L(x)}{\mu, A, P, lvl_a \vdash\ x := call\ B.f(\overline{e})}$

represents the security level of the expression after declassification, μ represents the current system state, c represents program statement and P represents the permission context. For example, when the system state is μ, a declassification statement $x :=\downarrow e$ is executed on the permission context P. At this time, its declassification predicate is $D(lvl_e, L(x), \mu, x :=\downarrow e, P)$.

Whether declassification predicate is hold depends on the permission function $\Gamma(e)$ of the declassification expression e. If the permission function $\Gamma(e)$ of the expression e is defined and $\Gamma(e) \subseteq P$, then the declassification predicate holds, and the declassification operation is secure at this time. Otherwise, either when the permission function $\Gamma(e)$ of the expression e is not defined, or the permission function is defined, but $\Gamma(e) \not\subseteq P$, the declassification predicate does not hold, and the declassification operation is insecure at this time.

D. Rules

Our proposed logic rules are shown in the table V.

Some of these rules, such as **R-Skip** and **R-Seq** statements are relatively simple, and we can be easily analogized to Hoare logic. For assignment statements, we can divide them into two cases **R-UAsgN** and **R-LAsgN** according to whether the variable defines the security level function L. If a variable x does not define the security level function $L(x)$, it means that the variable can receive the value of the expression of all security levels, that is, its security level is the highest. Therefore, in this case, it is secure to assign any expression to the variable x. When the security level function $L(x)$ of the variable x is defined, it means that the variable x can contain a security level that cannot exceed $L(x)$. At this time, the assignment statement needs to guarantee $lvl_e \leq L(x)$, otherwise we think that the information flow is insecure.

The conditional statement **R-If** is similar to the loop statement **R-While**. In order to prevent the attacker from obtaining information of high security level in the condition through different running results, we require that the security level of the attacker is not lower than the security level of the condition, that is $lvl_e \leq lvl_a$.

Local variable definition statement **R-DefL**, since this statement is not an assignment statement, we have no requirements for the definition of the variable, we only need to ensure that the variable is secure for information flow within its scope. The permission check statement **R-Check**, we divide the statement into two cases according to whether the atomic permission p belongs to the permission set P. Each situation corresponds to two different permission sets P. In addition, we also need to ensure that the information flow of the program is secure in each branch of the permission check.

Because we defined the declassification predicate D in the previous section, the declassification assignment statement **R-AsgD** is secure only if the declassification predicate holds, otherwise it is not. For the function call statement **R-Call**, we need to satisfy the information flow security inside the function body and the function return value respectively.

VI. Verification

In this section we illustrate the usefulness of our proposed logic through an example. The example code is shown in Table VI below.

We can think of this example as a mobile banking login application. When we log into the mobile banking, we must first perform mobile code verification on our mobile phone, and then we need to identify the person before we can enter the bank account. We can assume that the above code simulates this function, where the permission p_1 indicates whether the user has the permission to verify the mobile phone code verification, and the permission p indicates the permission of the identity verification. Obviously, these two permissions are indispensable, otherwise we will not be able to login in normally.

TABLE VI: An application example

```
B.funB(){
   init y=0;
   in{
     check(p1){
       then  y:=call A.fun();
⟨mem[y ↦ 0], tr⟩, B, P_B, lvl_a ⊢ y := callA.fun()
       else  y:=0;
     }
⟨mem[y ↦ 0], tr⟩, B, P_B, lvl_a ⊢ check(p1) then c1 else c2
     return y;
   }
μ, B, P_B, lvl_a ⊢ init y = 0 in c
}
A.fun(){
   init x=0;
   in{
     check(p){
       then  x:=↓ information;
⟨mem[y ↦ 0, x ↦ 0], tr⟩, A, P_B, lvl_a ⊢ x :=↓ information
       else  x:=0;
     }
⟨mem[y ↦ 0, x ↦ 0], tr⟩, A, P_B, lvl_a ⊢ check(p) then c1 else c2
     return x;
   }
⟨mem[y ↦ 0], tr⟩, A, P_B, lvl_a ⊢ init x = 0 in c
}
```

In this example, we assume that the permission set P_B of the application B contains the permissions p and p_1, then in the process of executing the application B, the permission context P is P_B. First we define a local variable y. Then execute the $check(p_1)$ statement. Because the permission set of the application B contains the permission p_1, the $then$ branch is executed to enter the function call statement. In the function call statement, we should use the permission set P_B of the application B as the permission context into the application A for execution. After defining the local variable x in A, enter the $check(p)$ statement. At this time, because the permission set P_B also contains the permission p, we use declassification assignment statement assigns the high security level $information$ to the variable x with the reduced security level, and returns it through the return value. After returning to the application B, we use the variable y to receive the return value of the function call, and get the final bank account information, and the login is successful.

In information flow security, the use of local variables needs to ensure that the information flow is secure in its scope, and the function call needs to ensure that the information flow is secure in the function body. Therefore, if and only if the security level of the return value of the function is less than or equal to $L(y)$ and the classification predicate $D(lvl_{information}, L(x), \mu, x :=\downarrow information, P)$ holds, the information flow is secure, otherwise it is insecure.

VII. Conclusion and Future Work

In this work, We present a formal language and the corresponding logical rules for proving the information flow security of mobile applications. Our approach has well defined semantics and makes use of a permission based declassification strategy, which makes the reasoning more accurate.

In the future, we would like to extend the access control policy to consider solutions to the branching problem that relies on secrets. In addition, we will also extend the semantics and the logic to handle the problem of non-monotonic of permissions.

Acknowledgements

This work is supported by Shanghai Science and Technology Commission Program under Grant 20511106002, Shanghai Trusted Industry Internet Software Collaborative Innovation Center and the Fundamental Research Funds for the Central Universities.

References

[1] M.Krohn and E.Tromer, "Noninterference for a practical difc-based operating system," in *Proceedings of the 2009 IEEE Symposium on Security and Privacy*, 2009, pp. 61–76.

[2] N. B. Said and L. Cristescu, "End-to-end information flow security for web services orchestration," *Science of Computer Programming,187:102376*, 2020.

[3] J. Bacon, D. Eyers, T. Pasquier, J. Singh, L. Papagiannis, and P. Piezuch, "Information flow control for secure cloud computing," in *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, 2014, pp. 76–89.

[4] "Review and prospect for information flow security technology," in *Journal of Nanjing University of Posts and Telecommunications*, vol. 31, no. 5, 2011.

[5] L. Mengjjun, L. Zhoujun, and C. Huowang, "A survey of security protocol verification base on process algebras," in *Journal of Computer Reserach and Development*, vol. 41, no. 7, 2004, pp. 1097–1103.

[6] D. Zhiyi, S. Guoxin, and S. Zhiqing, "Type system and the correctness of program," in *Computer Science*, vol. 33, no. 1, 2006, pp. 141–143.

[7] W. Libin, "Information flow control for integrity based on type system," in *Journal of South China Normal University*, vol. 3, 2006, pp. 42–47.

[8] L. Chengcheng, Z. Yongsheng, and L. Guangyu, "Research on a secure semantic web services mode," in *Computer Technology and Development*, vol. 20, no. 2, 2010, pp. 170–174.

[9] T. Murray, R. Sison, and K. Engelhardt, "Covern: A logic for compositional verification of information flow control," in *2018 IEEE European Symposium on Security and Privacy*, 2018, pp. 16–30.

[10] N. Coughlin and G. Smith, "Rely/guarantee reasoning for noninterference in non-blocking algorithms," in *2020 IEEE 33rd Computer Security Foundations Symposium(CSF)*, 2020, pp. 16–30.

[11] L. Hao, L. Qiang, Y. Jiwen, and Q. Peide, "A security system model based on mandatory access control and information flow," in *Computer Engineering and Science*, vol. 27, no. 3, 2005, pp. 16–20.

[12] D. Schoepe, T. Murray, and A. Sabelfeld, "Veronica:expressive and precise concurrent information flow security," in *IEEE 33rd Computer Security Foundation Symposium(CSF)*, 2020.

[13] J.Goguen and J.Mesegue, "Security policies and security models," in *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, 1982.

[14] R. Giacobazzi and I. Mastroeni, "Abstract non-interference: A unifying framework for weakening information-flow," in *ACM Transactions on Privacy and Security*, vol. 21, no. 2, 2018.

[15] A.Banerjee and D.A.Naumann, "Stack-based access control and secure information flow," in *Journal of Functional Prpgramming*, vol. 15, no. 2, 2005, pp. 131–177.

[16] H. Chen, A. Tiu, Z. Xu, and Y. Liu, "A permission-dependent type system for secure information flow analysis," in *IEEE 31st Computer Security Foundations Symposium*, 2018, pp. 218–232.

[17] Y. Zhao, X. Wu, J. Liu, and Y. Yang, "Formal modeling and security analysis for openflow-based networks," in *International Conference on Engineering of Complex Computer Systems*, 2018, pp. 201–204.

[18] Y. Zhao, X. Zhang, L. Shi, G. Zeng, F. Sheng, and S. Liu, "Towards a formal approach to defining and computing the complexity of component based software," in *Asia-Pacific Software Engineering Conference*, 2019, pp. 331–338.

Unlearnable Examples: Protecting Open-Source Software from Unauthorized Neural Code Learning

Zhenlan Ji, Pingchuan Ma, Shuai Wang
The Hong Kong University of Science and Technology
{zjiae,pmaab,shuaiw}@cse.ust.hk

***Abstract*—The vast volume of "free" code maintained on open-source code management systems significantly simplifies the process of producing and sharing open-source software. Recently, we have seen a growing trend in which these open-source software is being used for neural code learning without authorization. Note that open-source software does not necessarily imply "unrestricted usage," e.g., software under the BSD license requires users to retain the copyright notice and credit the software's developers.**

The unauthorized use of software for (commercial) neural code learning models has raised copyright concerns. This paper, for the first time, provides approaches for protecting open-source software from unauthorized neural code learning via unlearnable examples. Our proposed technique applies a set of lightweight transformations toward a program before it is open-source released. When these transformed programs are used to train models, they mislead the model into learning the unnecessary knowledge of programs, then fail the model to complete original programs. The transformation methods are sophisticatedly designed to ensure that they do not impair the general readability of protected programs, nor do they entail a huge cost. We focus on code autocompletion as a representative downstream task of unauthorized neural code learning. We demonstrate highly encouraging and cost-effective protection against neural code autocompletion.

I. Introduction

Recent advances in deep neural networks (DNNs) have resulted in advancements in computer vision (CV) and natural language processing (NLP) applications. Recently, there has been a surge of interest in using neural networks to solve a variety of software engineering (SE) tasks by learning from codes, including program synthesis [10], autocompletion [15], and code summarization [2]. For example, GitHub recently released Copilot [8], with the aim to provide an "AI pair programmer" capable of automatically generating programs from natural language specifications.

The major success of neural code learning is attributed in part to the availability of large-scale corpus [15]. For instance, Copilot is trained on massive amounts of open-source code, including public code collected from GitHub [3]. Nonetheless, a widespread concern is that some datasets were amassed without mutual consent [16]. In fact, it has been extensively noted that Copilot may leak sensitive code snippets when performing auto programming [23]. And, as its developers admit, Copilot uses all public Github code for training regardless of license [11], thus likely breaching the copyright of lots of open-source software. In this paper, we refer to the use of code as training data without consent as "*unauthorized utilization.*" Note that "open-source" software does **not** necessarily grant model owners like GitHub the right to sell the software content, nor does it grant model owners the right to distribute/use open-source software. In short, these neural code learning applications have raised primary concerns about unauthorized usages of open-source software.

There are solutions to protect private data against unauthorized machine learning. Recent work, in particular, generated unlearnable images via adversarial transformations [9], [24], [4], fooling the model into believing that there is nothing that can be learned from the transformed images. This work advocates for a focus on creating unlearnable examples of code that are difficult for neural code learning models to learn and, meanwhile, exhibit identical functionality and high readability to humans. However, in comparison to images, it is more difficult to practically enforce unlearnability on software with "adversarial transformations," because arbitrarily flipping a token in a program may change its semantics and even impose grammatical errors. Existing techniques (e.g., software obfuscation) used to protect software from exploitations are often heavyweight, which significantly impair the readability and is thus undesirable for open-source software.

In this paper, we design a set of lightweight *semantics-preserving* transformations toward software. Users who plan to open-source release their software can first locally launch our transformations toward their programs before releasing them. Given the inadequacy of a single transformation, we form an iterative transformation process and identify an optimal transformation sequence using multi-armed bandits. We employ a commonly-used code embedding model to guide local transformation, with the aim of maximizing the embedding distance between a transformed program and its clean version while preserving a modest edit distance (to retain readability). Users can then release the transformed program, and when the released program is used as training data for neural code learning (which is generally *not avoidable* for open-source software), these transformed programs deceive downstream applications like autocompletion by constantly learning faulty knowledge. Thus, the transformed program though remains accessible to the open-source community, becomes unlearnable from the perspective of unauthorized neural models.

We use CodeBERT [7] as the local model to guide the transformation. We deceive CodeGPT [12], a SOTA code

DOI reference number: 10.18293/SEKE2022-066

autocompletion model trained on the POJ-104 dataset [15] with one or several programs that have been transformed by our work. The evaluation shows that the transformed programs can achieve a high success rate of creating unlearnable examples, greatly reducing the accuracy of CodeGPT (to nearly "random"), while maintaining decent readability and a small runtime cost.

Contributions. We summarize our contributions as follows: 1) conceptually, we advocate for a new focus on protecting software against unauthorized neural code learning, a growing concern in the open-source community, 2) technically, we propose a set of lightweight transformations to transform software in a semantics- and readability-preserving manner. The optimal transformation sequence toward a program is determined using multi-armed bandits, and 3) we demonstrate empirically that transformed programs effectively mislead unauthorized neural code autocompletion with negligible cost.

Open Source. We release our code at https://github.com/ZhenlanJi/Unlearnable_Code.

II. Related Work

Mitigating Data Privacy Leakage. The majority of work on reducing model training data privacy leaks is based on federated learning [13]. Rather than sharing the training data, owners of training data share the model updates to jointly train a model. Contemporarily, differential privacy assures that a trained model does not learn raw training data [6]. In contrast, our work focuses on a more challenging scenario: protecting open-source software from being learned by unauthorized neural models. That is, the deep learning models are not trusted in our scenario, and it is unclear if they have utilized any privacy-preserving techniques like differential privacy. Recent efforts have been made to protect the privacy of photos with adversarial transformations against facial recognition [24], [4]. We protect open-source software, a timely albeit under-explored subject. We propose a general framework that produces unlearnable programs that can be used to mitigate unauthorized code embedding applications.

Protecting Unauthorized Code Usage. Some prior research focuses on identifying code authorship [1]. Nevertheless, protecting unauthorized usage of open-source software, though having raised widespread concern, is rarely discussed. One contemporary research inserts unusable code snippets ("deadcode") to impede unauthorized code usage [20]. Nonetheless, inserting deadcode, e.g., a special function that is never executed (referred to as "watermark" in their paper), is generally easy to be recognized and removed. Transformations proposed in this work are stealthy and hard to elide, meaning that our approach is more robust and effective in defeating unauthorized code usage.

III. Approach Overview

Research Challenge. Existing works generate "unlearnable" photos by applying adversarial transformations [24], [4]. These approaches, however, are inapplicable to software. Software is written in a structured manner, where transforming arbitrary bytes in a program can easily break its functionality. On the other end of the spectrum, software obfuscations techniques [5] transform software into an unreadable form, which may be used to defend unauthorized utilization. However, obfuscations can largely hamper software readability and distribution in the open-source community and consequently is *not* widely used. In sum, we aim to deliver a lightweight and effective method, such that software is transformed without impeding its functionality, retaining high readability, while making it "unlearnable."

Assumption and Objective. We aim to design a set of transformations, particularly toward software. These transformations change the program source code in a semantics-preserving manner while retaining readability and execution speed. This way, we protect open-source software against unauthorized neural code learning, which is jeopardizing copyright of open-source software for unauthorized usages.

While authors may have no direct access to the unauthorized neural code models, they can set up SOTA code embedding models like CodeBERT locally to guide transformations. However, once the transformed software is released as open-source, authors cannot prevent unauthorized usage, which will include the released software as part of their training data. We establish a practical objective for protection, such that the released software, after local transformation, cannot be correctly matched to its original version in front of representative neural code learning applications like code autocompletion. In the rest of the paper, we use "clean version" to refer to the original, untransformed program to ease the reading.

TABLE I
TRANSFORMATION METHODS.

Class	Methods	Abbreviations
Identifier Level	identifier replacement	IR
	identifier synonym	IS
	character synonym	CS
Constant Level	int rewrite	INT
	float rewrite	FLT
	string rewrite	STR
Statement Level	int def. insert	IDI
	float def. insert	FDI
	string def. insert	SDI
	single line insert	SLI

A. Semantics-Preserving transformation

This research designs a set of transformation methods for programs. Each method provides a *semantics-preserving* transformation, in the sense that the transformed output, another piece of software, retains the original functionality. We further clarify that all transformation methods are intended to provide *simple* and *incremental* transformations, while heavyweight transformations can undermine the readability of open-source software. Table I lists all designed transformation methods.

Identifier Level. We first parse a program to extract all identifiers (i.e., variable names). We then design three transformation methods on extracted identifiers. Identifier replacement (IR) randomly replaces a subset of identifiers with random

strings. Identifier synonym (IS) is built on the basis of WordNet [14], a large-scale lexical database. We query WordNet for a given identifier i to see if i and its synonyms exist; if so, we replace i with a randomly selected synonym. Compared with IR, IS can better preserve the readability of transformed programs. We also propose character synonym (CS) which uses hardcoded rules to replace a single-character identifier with another character, e.g., `i` $\rightarrow$ `j`.

As discussed in Sec. II, many code embedding models regard program statements as sentences, with identifiers (and also constants; see below) treated as tokens. This allows software to be smoothly processed via NLP techniques. Accordingly, we envision that identifier-level rewriting, though lightweight and retains readability to a large extent, will be useful in token-level transformation and deceiving neural models. This intuition is supported in our evaluation (Sec. V).

Constant Level. Additionally, we also propose three schemes to transform constants extracted in a program without breaking the semantics. Int rewrite (INT) converts an integer into an arithmetic expression. For instance, after applying IR, the C statement `a = a + 10` will be converted into `a = a + 121 - 21`, where "121" and "21" are two randomly-decided constants that will guarantee to yield "10". Similarly, we design a transformation scheme, namely float rewrite (FLT), to convert floating point numbers into arithmetic expressions. As for strings, our scheme string rewrite (STR) converts a `char*` constant in C (or `string` constant in C++) into two substrings, where the original string is recovered during runtime by calling the `libc` function `strdup` followed by calling another `libc` function `strcat`.

Existing DNN-based authorship identification and code clone detection gain from matching "magic numbers," which denote representative constants [18]. As a result, transforming constants should be effective in misleading the model. Moreover, as mentioned in Sec. II, some code embedding techniques learn the program abstract syntax tree (AST). Accordingly, by converting constants to expressions, we obscure the AST without largely impeding readability.

Statement Level. We also design methods to extend existing statements or insert new statements. For the IDI, FDI, and SDI schemes, we split an existing declaration statement of integer, float, or string variables into multiple declarations. For instance, IDI adds one extra statement with integer arithmetic computation, ensuring that the result is equal to the original declarations. Similarly, FDI and SDI insert new statements that perform floating-point arithmetics or string manipulations without changing the original functionality. In contrast, SLI generates and inserts new declarations at random, whose declared variables are never used in subsequent computation.

While the semantics is retained during transformation, the newly-inserted statements can complicate program AST and control flow graphs. We anticipate that neural code learning models that rely on AST or program structural-level information will struggle to understand the transformed programs.

Algorithm 1: Protection framework.

Input: Clean Program s, Transformation Sequence Length K, Sample Size M, Batch Size m, Exploration Factor ϵ, Discounting Factor γ
Output: transformed Program $\hat{s}$

1 $\hat{s} \leftarrow s$;
2 **foreach** $k \leftarrow 1, \cdots, K$ **do**
3 $\quad D \leftarrow [0]^{|\mathcal{T}|}$; // initialize expectation distribution
4 $\quad S \leftarrow \emptyset$; // transformed software set
5 $\quad$ **foreach** $t \leftarrow 1, \cdots, M/m$ **do**
6 $\quad\quad T \leftarrow \emptyset$;
7 $\quad\quad$ **foreach** $i \leftarrow 1, \cdots, m$ **do**
8 $\quad\quad\quad t \leftarrow$ `random()` with probability of $\gamma^{i-1}\epsilon$;
9 $\quad\quad\quad t \leftarrow \arg\max_t D_t$ with probability of $1 - \gamma^{i-1}\epsilon$;
10 $\quad\quad\quad T \leftarrow T \cup \{t\}$;
11 $\quad\quad$ **end**
12 $\quad\quad$ transform $\hat{s}$ with all $t \in T$ and update D;
13 $\quad\quad$ record all transformed program to S;
14 $\quad$ **end**
15 $\quad \hat{s} \leftarrow \arg\max_{s \in S} f(s)$;
16 **end**
17 **return** $\hat{s}$

IV. Framework Design

Motivation. Each transformation scheme complicates program structures to some degrees, and different schemes may achieve a *synergistic effect* by iteratively modifying a program, with the output of each iteration, a transformed program, serving input of the next iteration. As a result, the applied transformation sequences (we have ten transformation methods) create a vast search space, 10^K, where K represents the number of iterations applied to the input. We formulate searching for the optimal transformation sequence as an optimization process, where statistical methods can help to promptly explore the search space and find optimal sequences.

Framework. The input of our pipeline is a program s. Let the local embedding model (e.g., CodeBERT) be $\mathcal{E}$, we iteratively transform s into $\hat{s}$ until $\hat{s}$ manifests a large embedding distance and also a small edit distance with s. This way, the "identity" of s will be hidden, given that $\hat{s}$ is seen as irrelevant to s in the view of M while similar to s in view of humans (by edit distance). Users can then release $\hat{s}$. Soon we will show that $\hat{s}$ will be protected from disclosing the information of s even if code autocompletion models are trained over $\hat{s}$.

Alg. 1 illustrates the protection workflow. For each iteration, we search for an optimal transformed software with ϵ-Greedy (line 3–15) such that a predefined objective function is maximal. Note that since transforming software is stochastic (e.g., the choice of synonyms is random), the effectiveness of each transformation method forms a random variable. Therefore, it requires extensive sampling to derive an optimal transformed software. Our sampling strategy is largely enlightened by ϵ-Greedy in standard multi-armed bandit problem. In this setting, the sampling strategy is progressively refined and favors the transformation methods that have good historical performance. First, it samples a batch of transformation methods: it 1) takes a random method with the probability of ϵ (line 8) or 2) takes the best method (w.r.t. historical expectation on f) with the probability of

$1-\epsilon$ (line 9). Typically, the former case explores all possible transformation methods, while the latter seeks to maximize the expected value of f. Alg. 1 collects samples in T and transforms $\hat{s}$ with each $t \in T$ respectively, and updates the table of expected value per batch (line 12–13). We define the objective function f as:

$$f(\hat{s}; s) = \frac{\texttt{l2-norm}(\mathcal{E}(\hat{s}), \mathcal{E}(s))}{\texttt{edit-dist}(\hat{s}, s)} \quad (1)$$

where `l2-norm` measures the euclidean distance of transformed $\hat{s}$ and its clean version, while `edit-dist` measures the edit distance between them. By doing so, we presume Alg. 1 will gradually find $\hat{s}$ with sufficiently long euclidean distance while retaining a small edit distance with s.

Hyper-parameters. Alg. 1 takes hyper-parameters. K denotes the length of the transformation sequence (i.e., how many iterations are allowed to transform s); M denotes the sample size for deciding one single transformation method in ϵ−`Greedy`. m is defined as the batch size, where the distribution table is updated per batch (in contrast to per sample). ϵ is the exploration rate that balances exploration and exploitation in ϵ−`Greedy`, and γ is the discounting factor that reduces the probability of exploration when the distribution is well captured by prior trials. Overall, larger K and M indicate more intensive transformation. For the current implementation, we set $K = 15$ and $M = 256$. Our evaluation shows that this configuration enables a reasonable tradeoff between effectiveness and cost. Users are encouraged to configure these two hyper-parameters according to their own usage scenarios. For m, ϵ, and γ, they are all common settings for ϵ−`Greedy`, where $m = 64$, $\epsilon = 1$, and $\gamma = 0.5$ in our implementation.

V. Evaluation

Neural Embedding. We use `CodeBERT`, a SOTA code embedding model to guide our local transformation shown in Alg. 1. We have introduced the high-level concept behind `CodeBERT` in Sec. II. We emphasize, however, that our protection pipeline is *orthogonal* to specific embedding models used during local transformation.

Test Dataset. We use POJ-104 [15], a widely-used dataset containing 52,000 C/C++ programs written for 104 tasks. These programs implement programming assignments by students (e.g., two sum). While programs belonging to the same task share identical functionality, programs in different tasks are irrelevant. On average, each POJ-104 program contains about 36 lines of code (exclude white space and comments), whose length is comparable or outperforms the program datasets used by relevant works [21], [22], [25].

Code Autocompletion. We measure how the transformed programs can successfully mislead unauthorized code autocompletion models when being exposed. This is a timely topic, where modern code autocompletion tools like Copilot have raised concerns by training code on GitHub without distinguishing licenses. We use CodeGPT [12], a transformer-based code autocompletion model, for this task. CodeGPT extends the standard GPT-2 model structure and demonstrates that it performs at a SOTA level in this line of research [12].

Setup & Metrics. We measure when training CodeGPT using our transformed programs, whether the transformed program can be successfully protected from being used for *self-completion*. Let the training split of POJ-104 contain N programs, we measure three setups: randomly selecting $\{1, 20\%, 40\%\}$ programs and replacing them with their transformed versions using our approach. Note that "1" indicates that only one program is being transformed. We then train three CodeGPT models with these three training datasets. We consider two baselines: 1) B_1, which uses the original training dataset to train CodeGPT, and 2) B_2, which replaces those transformed $\{1, 20\%, 40\%\}$ programs with irrelevant POJ programs. Ideally, CodeGPT trained using our transformed programs would behave similarly to B_2 while deviating significantly from B_1, showing that the identities of protected programs have been successfully hidden.

To assess autocompletion, the standard approach randomly splits a program p into two pieces, p_1 and p_2, and uses p_1 as the model input to determine whether the model-generated piece p_2' matches p_2. Autocompletion models are typically assessed in terms of their one-line, three-line, and five-line completion accuracy, such that we match the first one line, three lines, and five lines of p_2' and p_2. To "match" two code snippets, both edit similarity (ES) and exact match (EM) are employed, with ES computing the tree edit similarity between the two code snippets (higher is better), and EM requiring an exact match between the two code snippets. We clarify that these two metrics are consistent used in measuring the performance of CoedXGLUE [12].

Model Training. Our learning and testing were conducted on a server machine with an Intel Xeon E5-2683 v4 CPU at 2.40 GHz with 256 GB of memory and two Nvidia 2080 GPU cards. The machine runs Ubuntu 18.04. Note that we use a pre-trained `CodeBERT` model to compute code embeddings. We share the same hyperparameters with CodeXGLUE to train the CodeGPT model. To benchmark the performance of our trained model, we also evaluate it on dataset py150 provided by CodeXGLUE. The result shows that our model has a comparative performance on py150 with CodeXGLUE's official report. The average training cost for each CodeGPT model is about five hours. It takes about one minute to transform one program using methods in Sec. IV.

A. Generating transformed Code Samples

TABLE II
AVERAGE EMBEDDING DISTANCE AND EDIT DISTANCE OF 1) TRANSFORMED PROGRAMS AND THEIR CLEAN VERSIONS; 2) SAME CLASS PROGRAMS; AND 3) CROSS CLASS PROGRAMS.

	transformed vs. Clean	Same Class	Cross Class
Embedding Dist.	3.54	3.54	4.29
Edit Dist.	138.2	480.7	587.8

transformation. We first compute and compare the average embedding distance and edit distance between transformed programs and corresponding clean versions in Table II. Furthermore, given that programs in POJ-104 are annotated with different classes, we also report pairwise distance among

programs of the same class or cross classes. Our approach effectively increases the difference between transformed and clean versions of the programs in CodeBERT's view, while retaining a reasonable edit distance.

The edit distance between the transformed and clean programs is much lower than that between programs of the same class. Programs from different classes implement distinct tasks, resulting in even longer edit distance. This reveals the high similarity in the view of users. We present further discussions on readability below in Table IV. In short, our transformation is shown to be effective in misleading neural code embedding, whose effectiveness will be further illustrated by defeating CodeGPT in Sec. V-B.

TABLE III
TRANSFORMATION DISTRIBUTION.

transformation	Portion	transformation	Portion
IR	26.6%	STR	6.3%
IS	3.1%	IDI	3.7%
CS	14.1%	FDI	0.1%
INT	29.3%	SDI	0.8%
FLT	1.1%	SLI	14.8%

Distribution of Applied transformations. As shown in Table I, we implement ten transformation methods. Recall that, in Alg. 1, a transformation method is retained, in case it effectively reduces embedding distance without primarily undermining the edit distance. Table III reports the distribution of successfully retained transformations. Overall, we interpret that all proposed methods (except FDI) are applied for a non-trivial amount of iterations. Identifier-level and constant-level transformations are particularly effective to deceive CodeBERT. To compute embeddings, CodeBERT extracts a large number of string and integer constants (including variable names) from input programs. Accordingly, by transforming identifiers and constants, CodeBERT can be effectively deceived. Note that the IS scheme is used less frequently. IS replaces a variable name with its synonym by querying WordNet. We find that a considerable fraction of variable names lack entries or synonyms in WordNet. Recall FLT extends a floating number into an arithmetic expression, and FDI rewrites a declaration statement for floating point variables. POJ-104 programs rarely use floating numbers. According to our observation, another reason that impedes the usage of statement-level transformations (IDI, FDI, SDI) is that they induce a higher edit distance, thereby undermining readability.

TABLE IV
AVERAGE SIMILARITY DECIDED BY JPLAG COMPARING TRANSFORMED PROGRAMS AND THEIR CLEAN VERSIONS.

transformed vs. Clean	Same Class	Cross Class
67.5%	4.4%	0.4%

Readability. The "readability" of software is often subjective and difficult to quantify. In addition to the edit distance, which our pipeline optimizes for, we provide another metric for the readability of transformed programs using conventional code similarity analyzers. At this point, we use a popular similarity checker called JPlag [17]. JPlag is widely used to detect code plagiarism based on syntactic and code structure-level features. Thus, using JPlag to evaluate code similarity provides a more complete picture of the readability of transformed code. JPlag can be configured locally prior to use. We also looked at another well-known tool, Moss [19]. Nonetheless, its remote server frequently fails to respond.

We assess randomly selected samples within POJ-104 as a baseline (same setting as Table II). We find that when randomly selected code samples from different classes are compared, the baseline similarity between them is only 0.4%, demonstrating that JPlag is capable of successfully distinguishing distinct programs. Note that JPlag can also distinguish programs belonging to the same class (for example, two quick sort programs) as long as their implementations are sufficiently distinct (average similarity 4.4%). In contrast, transformed programs exhibit a high degree of similarity to their clean versions, with an average similarity score of 67.5%. As a result, we interpret that these transformations do not primarily harm open-source software's "readability" and disseminability.

Cost. Our proposed transformations are functionality preserving, meaning that the transformed programs manifest identical functionality with their clean versions by design. Nevertheless, our transformations insert new code fragments into the programs, imposing additional performance penalty. We manually write non-trivial test cases for POJ-104 programs to assess performance penalty. We use a common performance analysis tool on Linux, perf, to measure the cost of (transformed) programs. In general, we report that the transformed programs become *negligibly* slow (on average less than 1%), if at all detectable. This is not surprising, as our methods primarily change symbols (variable names) and statements in a lightweight manner. Symbols are removed during compilation and hence have no effect on execution. In terms of statement-level changes, we find that many of them are optimized out during compilation. We interpret the cost evaluation as encouraging, illustrating that our transformation would incur negligible extra cost.

TABLE V
ASSESSING THE PERFORMANCE OF CODEGPT. B_1 AND B_2 ARE TWO BASELINE SETTINGS CLARIFIED IN THE **SETUP & METRICS** PARAGRAPH. UE (UNLEARNABLE EXAMPLE) DENOTES OUR RESULTS. "ES" STANDS FOR EDIT DISTANCE (A MORE TOLERATE METRIC OF CODE SIMILARITY), WHEREAS "EM" STANDS FOR EXACT MATCH. $B_2(20\%, 40\%, 1)$ AND UE $(20\%, 40\%, 1)$ CORRESPOND TO REPLACE $20\%, 40\%, 1$ OF CLEAN TRAINING PROGRAMS WITH OUR TRANSFORMED VERSIONS. UE IS EXPECTED TO CLOSE TO B_2 FOR HIGH PROTECTION ABILITY.

Method	1 line		3 lines		5 lines	
	ES	EM	ES	EM	ES	EM
B_1	73.9%	42.4%	71.2%	21.4%	68.3%	10.9%
$B_2(20\%)$	68.8%	34.9%	67.3%	15.9%	64.9%	7.2%
UE (20%)	69.3%	35.5%	67.7%	16.2%	65.2%	7.6%
$B_2(40\%)$	68.6%	34.5%	66.7%	15.6%	64.3%	6.7%
UE (40%)	68.4%	33.6%	66.7%	15.6%	64.2%	7.0%
$B_2(1)$	74.3%	48.8%	65.8%	10.2%	62.7%	0.9%
UE (1)	73.9%	45.8%	65.7%	10.8%	63.6%	2.2%

B. Mitigating Code Autocompletion

Table V reports the results of mitigating CodeGPT in different settings. When more lines are checked, CodeGPT's accuracy decreases. This is reasonable, as matching three or five lines implies a more difficult task than matching one line.

Recall that B_1 feeds CodeGPT with programs in its training dataset for self-completion, denoting the "upper bound" of accuracy. Table V reports promising results where the protected programs (three "UE" rows) are far from the B_1. More importantly, the "UE" rows are extremely close to their corresponding B_2 rows. As previously clarified, Baseline$_2$ denotes the "lower bound" of accuracy, as it replaces the transformed programs in the training dataset with irrelevant programs. Therefore, the evaluation results illustrate that, after transformation, protected programs behave similarly to randomly-picked programs, successfully deceiving the autocompletion model.

The 8th and 9th rows ($B_2(1)$ and UE (1)) denote inserting only *one* transformed program in the training dataset and feeding its clean version to CodeGPT (as noted in **Setup & Metrics**, we will randomly cut the input program into two and feed CodeGPT with the upper cut). This is a realistic setting, given that authors may want to protect their own piece of software before uploading it to GitHub. Though only changing one piece of training data, it already largely undermines the accuracy of CodeGPT when performing autocompletion toward the upper half of its clean version.

Clarification. We also clarify that the accuracy of CodeGPT in matching other programs (which may be also within the training data) has only negligible change (around 1%; particularly for the evaluation setting where 40% training data are transformed) compared with B_1. In summary, though it is generally hard (if at all avoidable) to prevent open-source software from being used as training data, information regarding the open-source software will not be leaked via autocompletion after applying our protection.

Case study. We provide an code example to illustrate the effectiveness of our approach to impede autocompletion at https://github.com/ZhenlanJi/Unlearnable_Code/blob/main/example.pdf.

VI. Conclusion

In this paper, we propose to mitigate unauthorized neural code learning from using open-source software. We design a set of lightweight transformations and explore optimal transformation sequences using multi-armed bandits. Our evaluation demonstrates that transformed programs can successfully deceive a SOTA code autocompletion model, CodeGPT.

Acknowledgement

We are grateful to the anonymous reviewers for their valuable comments. The work is supported in part by NVIDIA Academic Hardware Grant Program. Shuai Wang is the corresponding author of this paper.

References

[1] Mohammed Abuhamad, Tamer AbuHmed, Aziz Mohaisen, and DaeHun Nyang. Large-scale and language-oblivious code authorship identification. In *CCS*, 2018.

[2] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. ICLR.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] Valeriia Cherepanova, Micah Goldblum, Harrison Foley, Shiyuan Duan, John Dickerson, Gavin Taylor, and Tom Goldstein. LowKey: leveraging adversarial attacks to protect social media users from facial recognition. *arXiv preprint arXiv:2101.07922*, 2021.

[5] Christian Collberg, Clark Thomborson, and Douglas Low. A taxonomy of obfuscating transformations. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 1997.

[6] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[7] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. CodeBERT: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[8] GitHub. Copilot, 2021.

[9] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*, 2021.

[10] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 2017.

[11] Andrew Liu. Copilot, 2021.

[12] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[14] George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 1995.

[15] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. Convolutional neural networks over tree structures for programming language processing. In *AAAI*, 2016.

[16] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. An empirical cybersecurity evaluation of github copilot's code contributions. *arXiv preprint arXiv:2108.09293*.

[17] Lutz Prechelt, Guido Malpohl, Michael Philippsen, et al. Finding plagiarisms among a set of programs with JPlag. *J. UCS*, 8(11):1016.

[18] Erwin Quiring, Alwin Maier, and Konrad Rieck. Misleading authorship attribution of source code using adversarial learning. In *USENIX Security*, 2019.

[19] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *SIGMOD*, 2003.

[20] Zhensu Sun, Xiaoning Du, Fu Song, Mingze Ni, and Li Li. Coprotector: Protect open-source code against unauthorized training usage with data poisoning. *arXiv preprint arXiv:2110.12925*, 2021.

[21] Jeffrey Svajlenko, Judith F Islam, Iman Keivanloo, Chanchal K Roy, and Mohammad Mamun Mia. Towards a big data curated benchmark of inter-project code clones. In *ICSME*, 2014.

[22] Jeffrey Svajlenko and Chanchal K Roy. Evaluating clone detection tools with bigclonebench. In *ICSME*, 2015.

[23] Jake Williams. Copilot privacy leakage, 2021.

[24] Xiao Yang, Yinpeng Dong, Tianyu Pang, Hang Su, Jun Zhu, Yuefeng Chen, and Hui Xue. Towards face encryption by generating adversarial identity masks. In *ICCV*, 2021.

[25] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *arXiv preprint arXiv:1909.03496*, 2019.

DeepController: Feedback-Directed Fuzzing for Deep Learning Systems

Hepeng Dai [1] Chang-ai Sun [1 *] Huai Liu [2]

[1] University of Science and Technology Beijing
Email: daihepeng@sina.cn, casun@ustb.edu.cn

[2] Swinburne University of Technology
Email: hliu@swin.edu.au

Abstract

Deep learning (DL) systems are increasingly adopted in various fields, while fatal failures are still inevitable in them. One mainstream testing approach for DL is fuzzing, which can generate a large amount of semi-random yet syntactically valid test cases. Previous studies on fuzzing are mainly focused on selecting "quality" seeds or using "good" mutation strategies. In this paper, we attempt to improve the performance of fuzzing from a different perspective. A new fuzzer, namely DeepController, is accordingly developed, which makes use of the feedback information obtained in the test execution process to dynamically select seeds and mutation strategies. DeepController is evaluated through empirical studies on three datasets and eight DL models. The experimental results show that, with the same number of seeds, DeepController can generate more adversarial inputs and achieve higher neuron coverage than the state-of-the-art testing techniques for DL systems.

***Keywords**: Fuzzing, Deep Learning Systems, Software Testing*

1 Introduction

Nowadays, deep learning (DL) systems are used in a wide variety of fields, thanks to their powerful learning and reasoning capabilities. Nevertheless, DL systems, like traditional software systems, unavoidably contain some faults and thus show incorrect or unexpected behaviors. Testing is a main approach to support quality assurance of software systems. However, the unique features of DL systems pose new challenges for testing. For example, the logic behind a DL system is not manifested as that in traditional software; DL introduces much higher non-determinism in the software output. As such, many traditional testing techniques are no longer applicable to DL systems.

Among recently proposed testing techniques for DL systems, fuzz testing (or simply *fuzzing*) [1] is a basic technique that has gradually become a standard method in the industry. It can generate lots of semi-random test cases based on existing test data with relatively low cost. Despite the simplicity in concept, fuzzing has successfully generated the *adversarial inputs* that facilitate the fault detection [2]. Since the first white-box-based fuzzing method for DL systems [1], various fuzzing techniques have been proposed [3]. Some fuzzing techniques are focused on selecting appropriate seeds [1, 4], while others attempt to improve the performance of fuzzing via choosing "good" mutation strategies [5, 6].

Most previous studies on seed selection mainly used single pieces of test information (e.g., coverage information, the number of seed mutations, or the times of seeds added to the seed queue), but did not consider a variety of information comprehensively. Previous studies on mutation strategies showed that they tend to activate different sets of neurons [7], implying the uncertainty on the optimal mutation strategies for a specified seed during testing. Unfortunately, existing mutation strategy selection approaches did not individually consider specific seeds, which may affect the fault-detection efficiency of fuzzing.

In this paper, we propose a new fuzzing technique, namely *DeepController*, which makes full use of the feedback information (including coverage information, testing results, the number of seed mutations, and the times of seeds added to the seed queue) collected during the test execution process to guide the selections of seeds and mutation strategies. In line with software cybernetics [8], DeepController treats the whole testing procedure as a feedback control system, providing feedback-directed strategies for selecting seeds from the seed queue and choosing the most appropriate mutation strategies for selected seeds.

Our study has the following three major contributions:

(1) A comprehensive framework (Section 3.1), was proposed to implement DeepController, which can select seeds and mutation strategies that both have high potentials of fault detection, adaptive to the feedback information from test executions.

(2) Two algorithms, namely AS2 (Section 3.2) and AMS2 (Section 3.3), were developed for the selections of seeds and mutation strategies, respectively.

(3) A series of empirical studies on three datasets, and

*corresponding author

DOI reference number: 10.18293/SEKE2022-126

eight DL models (Section 4), were conducted to evaluate the performance of DeepController. As observed from these experiments, DeepController could generate more adversarial inputs, and obtain higher neuron coverage than the state-of-the-art DL testing techniques.

2 Background and Related Work

2.1 Fuzzing

The key idea of fuzzing is to generate a large amount of semi-random yet syntactically valid test cases by mutating existing test cases. Generally, the test case generation is composed of the following three components.

- *Seed queue construction*, which is responsible for constructing the seed queue by selecting test cases from corpus that contains a test pool as well as the label and coverage information of each test case.
- *Seed selection*, which is responsible to select seeds from the seed queue based on a certain selection strategy.
- *Seed mutation*, which uses some seed mutation strategies to generate mutated seeds that serve as test cases for executing the program under test (PUT).

2.2 Related Work

Recently, researchers have proposed a large number of fuzzing techniques for DL systems based on different theories and observations. Among them, coverage-guided fuzzing techniques have been proven to be very effective in detecting faults and exploring the internal states of the DL models. Closely related works are described below.

Seed Selection. Fuzzing iteratively selects seeds from the seed queue based on some strategies. One simple and commonly used strategy is random selection, but it does not use any information of testing process or DL systems. To enhance the random strategy, the idea of "recency-aware" was used to select the seed that induces the new coverage [9, 7, 4, 10, 11, 12], which corresponds to the observation that if a seed covers a branch, the following branches are more likely to be covered due to the hierarchical relationship between branches. In addition, the idea of "frequency-aware" was used to probabilistically select a seed based on the number of times it has been mutated: if a seed has already been picked many times, it has a lower probability of being selected again [4, 10].

Seed Mutation. As one core component of fuzzing, mutation strategy directly affects the fault-detection efficiency and effectiveness of fuzzing. The existing strategies for seed mutation can be classified into three categories: (1) gradient-based mutation strategies, which first calculate the gradient of the objective function, and then mutate the seed according to the calculated gradient [1, 13, 14]; (2) domain-knowledge-based mutation strategies, which mutate the seeds according to the properties of inputs, while the mutated seed has the same semantic information as the original one [7, 15, 16]; (3) search-based mutation strategies, which mutate the selected seeds using some search-based algorithms, such as population-based metaheuristics [5] and Monte Carlo Tree Search (MCTS) [6].

Although existing fuzzers showed promising results in detecting faults in DL systems, they do not make full use of the information obtained in test execution process, which could be useful for improving the efficiency of fuzzing, thus motivating this study.

3 Methodology

In order to further improve the performance of fuzzing for DL systems, this study makes use of the feedback information obtained in test executions to guide the selection of appropriate seeds and proper mutation strategies, particularly focused on designing new framework and two strategies for seed and mutation strategy selections.

3.1 Framework

Based on the principles of software cybernetics and the features of fuzzing, we propose the framework of DeepController, as illustrated in Figure 1. The starting point of DeepController is that there already exist a test suite and some mutation strategies. Note that researchers have devised many mutation strategies for different types of DL models, so it remains a challenging issue how to select them for testing. There is a feedback loop in the framework, which consists of basic components of fuzzing, DL model, the database for storing test information (including coverage information, the testing results, the times of the seed being selected, and the time of the seed adding to seed queue), and the controller for selecting seed and mutation strategy. Particularly, in the controller, the historical test information is leveraged to guide the selections of seeds and mutation strategies. Furthermore, the historical information can also be used to improve the underlying testing strategies.

3.2 Seed Selection Strategy

Suppose that a seed queue Q has n seeds, that is, $Q = \{s_1, s_2, \ldots, s_n\}$. $L_i(i = 1, 2, \ldots, n) = \{(s_i, t_0)\}$ is to store the time of s_i and the seeds generated based on s_i added to Q. For instance, seeds s_1^* and s_2^* are generated at time t_1^* and t_2^* by seeds s_1 and s_2, respectively. Both s_1^* and s_2^* can trigger some new coverage (e.g. new neurons). Then L_1 and L_2 should be updated as $L_1 = L_1 \cup \{(s_1^*, t_1^*)\}$ and $L_2 = L_2 \cup \{(s_2^*, t_2^*)\}$. The selection probabilities of L_i are denoted as $LP = \{\langle L_1, p_1\rangle, \langle L_2, p_2\rangle, \ldots, \langle L_n, p_n\rangle\}$, where $p_i(i = 1, \ldots, n)$ denotes the selection probability of L_i. There are two lists $E = \{e_1, e_2, \ldots, e_n\}$ and $E' = \{e_1', e_2', \ldots, e_n'\}$, where $e_i(i = 1, 2, \ldots, n)$ records the times the seeds in L_i are selected and trigger new coverage, while e_i' records the times the seeds in L_i are selected but no new coverage is triggered.

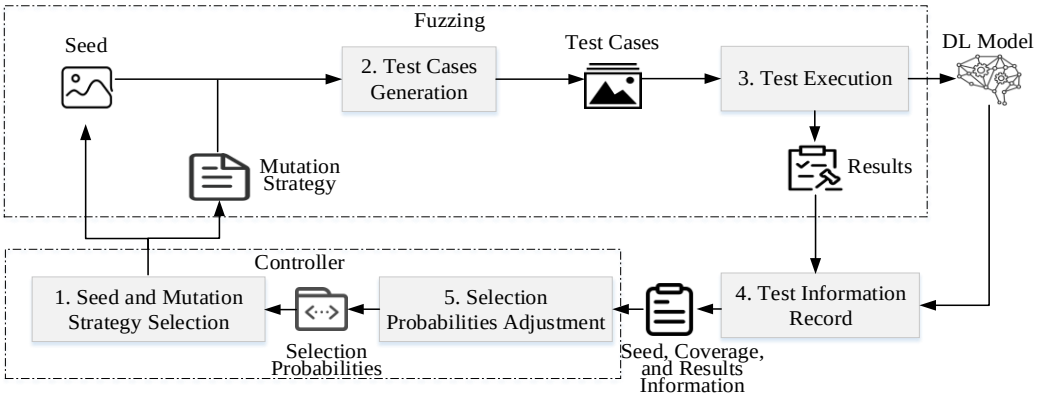


Figure 1: The framework of DeepController

We propose an adaptive seed selection strategy, namely AS2, which utilizes test information (including the coverage, the times of seeds being used, and the time of a seed added to seed queue) to select those seeds with higher fault-detection potentials. If the mutated seeds generated by s_i (that belonged to L_i) could trigger new coverage, the corresponding selection probability p_i of L_i will be increased; otherwise, p_i will be decreased. Moreover, the smaller the times of L_i being selected, the increase of p_i is greater; otherwise, the increase of p_i is smaller.

At the beginning, AS2 initializes $LP = \{\langle L_1, 1/n\rangle, \langle L_2, 1/n\rangle, \ldots, \langle L_n, 1/n\rangle\}$, and $e_i = 0, e'_i = 0 (i = 1, \ldots, n)$. During the test process, assume that AS2 selects L_i based on the LP, and the seed s_i that is the latest seed of L_i is selected. Accordingly, a set of mutated seeds $T = \{s_i^1, s_i^2, \ldots, s_i^k\}$ (k is the number of times a seed can be mutated) are generated based on s_i and some mutation strategies. Suppose that all seeds in T are executed. If $\exists s_i* \in T$, and s_i* triggers new coverage and $\forall j = 1, \ldots, n, j \neq i$, we then update $e_i = e_i + 1$ and set

$$p'_j = \begin{cases} p_j - \dfrac{\epsilon \times (1 + ln(1 + 1/(e_j + 1)))}{n - 1} & \text{if } p_j \geq W \\ 0 & \text{if } p_j < W \end{cases}, \tag{1}$$

where ϵ is a probability adjusting factor, and $W = \dfrac{\epsilon \times (1 + ln(1 + 1/(e_j + 1)))}{n - 1}$. Then,

$$p'_i = 1 - \sum_{j=1, j\neq i}^{n} p'_j. \tag{2}$$

Alternatively, If $\forall s_i* \in T$, and s_i* cannot trigger new coverage, we then update $e'_i = e'_i + 1$ and set

$$p'_i = \begin{cases} p_i - \epsilon \times (1 + e'_i/n) & \text{if } p_i \geq \epsilon \times (1 + e'_i/n) \\ 0 & \text{if } p_i < \epsilon \times (1 + e'_i/n) \end{cases}, \tag{3}$$

$$p'_j = \begin{cases} p_j + \dfrac{\epsilon \times (1 + e'_i/n)}{n - 1} & \text{if } p_i \geq \epsilon \times (1 + e'_i/n) \\ p_j + \dfrac{p_i}{n - 1} & \text{if } p_i < \epsilon \times (1 + e'_i/n) \end{cases} \tag{4}$$

.

AS2 keeps updating the selection probabilities of seeds via the formulas 1 to 4. As a result, the seeds with higher chances of triggering new coverage and being used less times have higher probabilities of being selected.

3.3 Mutation Strategy Selection Strategy

Suppose that there exist a seed queue $Q = \{s_1, s_2, \ldots, s_n\}$ and a set of mutation strategies $M = \{m_1, m_2, \ldots, m_x\}$. The list $MP_i (i = 1, 2, \ldots, n) = \{\langle m_1, p_i^1\rangle, \langle m_2, p_i^2\rangle, \ldots, \langle m_x, p_i^x\rangle\}$ can be created as the set of mutation strategy selection probabilities for each seed in Q. The list $C_i = \{c_i^1, c_i^2, \ldots, c_i^x\}$ records the times each strategy $m_h (h = 1, \ldots, x)$ has been used by s_i and triggered new coverage.

We propose an adaptive mutation strategy selection approach, namely AMS2, which analyzes the performance of mutation strategies on different seeds and selects the most appropriate strategy for a specific seed. If the mutated seeds generated by seed s_i and the mutation strategy $m_h (h = 1, 2, \ldots, x)$ could trigger new coverage or detect faults, the corresponding selection probability p_i^h of m_h will be increased; otherwise, p_i^h will be decreased. Moreover, the smaller the times of m_h being used, the increase of p_i^h is greater; otherwise, the increase of p_i^h is smaller.

At the beginning, AMS2 initializes $MP_i = \{\langle m_1, 1/x\rangle, \langle m_2, 1/x\rangle, \ldots, \langle m_x, 1/x\rangle\}$, and sets all elements in C_i to 0. During the test process, assume that s_i is selected by AS2. AMS2 selects a mutation strategy m_h based on the MP_i. Accordingly, a set of mutated seeds $T = \{s_i^1, s_i^2, \ldots, s_i^k\}$ (k is the number of times a seed can be mutated) are generated based on the s_i and m_h. Suppose

that all seeds in T are executed. If $\exists s_i* \in T$, and s_i* triggers new coverage or detects a fault, $\forall y = 1, 2, \ldots, x$ and $y \neq h$, we then update $c_i^h = c_i^h + 1$ and set

$$p_i^{y\prime} = \begin{cases} p_i^y - \dfrac{\delta \times x}{c_i^y + 1} & \text{if } p_i^y \geq \dfrac{\delta \times x}{c_i^y + 1} \\ 0 & \text{if } p_i^y < \dfrac{\delta \times x}{c_i^y + 1} \end{cases}, \tag{5}$$

where δ is a probability adjusting factor. Then,

$$p_i^{h\prime} = 1 - \sum_{y=1, y\neq h}^{x} p_i^{y\prime}. \tag{6}$$

Alternatively, If $\forall s_i* \in T$, and s_i* cannot trigger new coverage or detect a fault, we then set

$$p_i^{h\prime} = \begin{cases} p_i^h - \delta & \text{if } p_i^h \geq \delta \\ 0 & \text{if } p_i^h < \delta \end{cases}, \tag{7}$$

$$p_i^{y\prime} = \begin{cases} p_i^y + \dfrac{\delta}{x-1} & \text{if } p_i^h \geq \delta \\ p_i^y + \dfrac{p_i^h}{x-1} & \text{if } p_i^h < \delta \end{cases}. \tag{8}$$

AMS2 dynamically adjusts the selection probabilities of mutation strategies based on the formulas 5 to 8. As a result, the mutation strategies with a higher probabilities of triggering new coverage and detecting faults for the selected seeds have higher probabilities of being selected.

4 Empirical Study

We conducted a series of empirical studies to evaluate the performance of DeepController.

4.1 Research Questions

In our experiments, we focused on addressing the following three research questions.

RQ1 Can DeepController generate more adversarial inputs than the commonly used testing techniques for DL models?

RQ2 Can DeepController achieve higher neuron coverage (NC) [1] than state-of-the-art techniques?

RQ3 How about the performance of DeepController in terms of time overhead?

4.2 Experimental Design

DeepController was implemented as a self-contained fuzzing framework, written in Python based on the DL framework Keras (ver.2.1.6) with TensorFlow (ver.1.5.0) backend. With DeepController, we performed a comparative study to answer the three research questions raised above.

Datasets, DNN Models, and Baselines. We selected three popular publicly available datasets (i.e., MNIST [17], ImageNet [18], and CIFAR10 [19]) as the evaluation subjects (see Table 1). We further utilized the commonly used DL models, including LeNet-1, LeNet-4, LeNet-5, VGG-16 and VGG-19 for ImageNet, VGG-16 and VGG-19 for CIFAR10, and 20 layer CNN with max-pooling and dropout layers [6]. Note that most used models are open-source available, except VGG-16 and VGG-19 for CIFAR10, which were trained by ourselves. As summarized in a survey [3], there are several open-source tools for DL systems. To further measure the fault-detection ability of DeepController, we selected three representative fuzzers (DeepXplore [1], DeepTest [7], and DeepHunter [4]) and a gradient-based testing approach proposed recently (FGSM [20]) as our baselines.

Table 1: Subject datasets and DL models

Dataset	DL Model	Number of Parameters	Acc.(%)
ImageNet	VGG-16	138,357,544	92.60
	VGG-19	143,667,240	92.70
MNIST	LetNet-1	7,206	98.25
	LetNet-4	69,362	98.75
	LetNet-5	107,786	98.63
CIFAR10	VGG-16	138,357,544	86.84
	VGG-19	143,667,240	77.26
	CNN-20	952,234	77.68

Mutation Strategies. We select eight mutation strategies for image, which can be partitioned into two categories: (1) *Pixel Value Mutation* P, which includes image contrast, image brightness, image blur, and image noise; (2) *Affine Mutation* G, which includes image translation, image scaling, image shearing, and image rotation. The empirical results in [7] showed that combining different image mutation strategies, neuron coverage can be improved. In order to keep the semantics of the mutated seeds close to the original one, we adopt a conservative strategy that selects a pixel value mutation strategy p from P by using AMS2, and randomly selects an Affine mutation g from G. Then the mutated seeds could be obtained by applying selected p and g on the selected seed. Note that this study aims to improve the fault-detecting efficiency of fuzzing, the strategy proposed in [4] was used to judge whether the mutated seeds are valid or not.

Parameter Settings. The hyper-parameters of DeepXplore, DeepTest, DeepHunter, and FGSM were configured based on the settings in their original studies. For the probability adjusting factors ϵ and δ of DeepController, we conducted a series of trial experiments to find a fair setting, and finally set $\epsilon = 0.01$ and $\delta = 0.1$.

To reduce the randomness effect of experiments, we randomly generated ten seed queues for each dataset using different random seeds (each seed queue has 1000 images) and averaged the results. The termination condition of testing

Table 2: Average number of adversarial inputs generated by fuzzers on different datasets

Fuzzers	MNIST			ImageNet		CIFAR10		
	LeNet-1	LeNet-4	LeNet-5	VGG-16	VGG-19	VGG-16	VGG-19	CNN-20
DeepXplore	23.4	32.8	23.2	118.0	124.8	88.2	92.2	103.0
DeepTest	21.8	26.4	20.2	130.2	166.4	302.0	309.6	370.4
DeepHunter	24.8	24.4	38.6	132.6	150.4	381.4	308.2	472.6
FGSM	27.2	19.8	19.4	30.8	31.0	324.4	311.4	595.2
DeepController	**37.8**	**35.4**	**43.4**	**162.6**	**199.0**	**478.8**	**447.4**	**697.2**

Table 3: Average neuron coverage of fuzzers on different datasets

Fuzzers	MNIST			ImageNet		CIFAR10		
	LeNet-1	LeNet-4	LeNet-5	VGG-16	VGG-19	VGG-16	VGG-19	CNN-20
DeepXplore	44.62%	55.81%	53.88%	23.66%	22.20%	5.53%	5.37%	32.55%
DeepTest	47.88%	64.32%	55.60%	32.33%	29.25%	5.74%	5.24%	34.56%
DeepHunter	46.62%	63.86%	58.58%	26.98%	24.04%	5.47%	4.85%	34.26%
FGSM	38.46%	61.08%	57.39%	32.02%	28.95%	5.06%	4.76%	33.26%
DeepController	**48.08%**	**64.59%**	**59.11%**	**32.94%**	**29.80%**	**5.88%**	**5.57%**	**34.71%**

DL models on MNIST and CIFAR10 is the generation of 1000 test cases. Since testing the VGG models on ImageNet required huge testing resources, we set the generation of 500 test cases as the termination condition. Besides, in all experiments, we set the threshold of NC to 0.75 and the experiments are preformed on an Mac machine (one Intel i5 3733 MHz processor with four cores, 16GB of memory).

4.3 Results

RQ1: Generation of Adversarial Inputs. Table 2 reports the average number of adversarial inputs generated by different fuzzers. It is clearly shown that DeepController generated more adversarial inputs than the four baseline techniques.

We also conducted statistical testing to verify the significance of this evaluation. We used ANOVA [21] (with significance level $\alpha = 0.05$) to determine which pairs of testing techniques had significant differences. Our calculated results (the *f-ratio* value was 6.0951, and the *p-value* was 0.0001) show that DeepController was significantly better than the four baseline techniques in terms of the capabilities of generating adversarial inputs.

Answer to RQ1: DeepController generated more adversarial inputs than the four baseline techniques .

RQ2: Neurons Coverage. Table 3 reports the average neuron coverage achieved by the four baseline techniques and DeepController on different datasets. It is clearly shown that DeepController achieved higher neuron coverage on different models than the four baseline techniques.

Answer to RQ2: DeepController covered more neurons than the four baseline techniques.

RQ3: Time Overhead. DeepController makes use of feedback information to select appropriate seeds and mutation strategies, which might result in longer computation time as compared with DeepXplore, DeepTest, DeepHunter, and FGSM. The time overhead is mainly composed of the following: (1) *seed and mutation strategy selection time* that refers to how long it takes to select seeds and mutation strategies; (2) *seed mutation time* that refers to how long it takes to generate mutated seeds based on the selected mutation strategies and seeds; (3) *seed execution time* that represents the time required for executing mutated seeds. Table 4 reports the time overhead of the different techniques, where $x/y/z$ denotes the average time overheads of studied techniques on MNIST, ImageNet, and CIFAR10, respectively.

DeepController needed more time to select seeds. However, seed selection is an inexpensive process. On average, seed selection time was only 1.2% of the whole testing time. In terms of the whole test overhead, DeepController did not always have the longest testing time. Specifically, DeepController had shorter testing time than DeepXplore and FGSM on all datasets, shorter than DeepHunter on MNIST and CIFAR10, DeepTest had shorter testing time than DeepController on all datasets, but the difference was marginal.

Answer to RQ3: For executing the same number of test cases, DeepController generally had shorter testing time than DeepXplore, FGSM, and DeepHunter, but had marginally higher time overhead than DeepTest.

Table 4: Time overhead of the different techniques on different datasets

Techniques	Seed and Mutation Strategy Selection (s)	Seed Mutation (s)	Seed Execution (s)	Total (s)
DeepXplore	0.003/0.043/0.006	170.467/994.904 /596.079	4.658/3223.119/230.186	175.129/4218.066/826.271
DeepTest	0.003/0.015/0.053	0.384 /7.643 /0.447	1.817/1443.383/21.819	**2.204 /1451.041/22.319**
DeepHunter	0.880/0.150/0.147	157.648/31.388 /51.072	1.666/1440.198/21.035	160.194/1471.736/72.253
FGSM	0.002/0.013/0.020	4.048 /1497.753/131.868	1.681/441.829 /60.696	5.731 /1939.595/192.585
DeepController	0.782/0.087/0.151	2.204 /11.068 /1.676	1.790/1483.132/22.342	**4.776 /1494.288/24.169**

5 Conclusions and Future Work

Fuzzing has increasingly been proven to be very effective in detecting faults and exploring the internal states of the DL models. In recent years, researchers have proposed quite a few fuzzers. Nevertheless, the execution information has not been fully utilized in the state-of-the-art fuzzing techniques. In this paper, we introduced feedback into fuzzing for DL models, and proposed *DeepController*, which makes use of feedback information obtained in test executions to guide the selection of seeds and mutation strategies. Empirical studies were conducted to evaluate the performance of DeepController, in comparison with four popular techniques for testing DL models, based on three commonly used datasets, and eight DL models. The experimental results showed that the proposed approach can generate more adversarial inputs and explore more internal states of DL models, with at least similar time overhead.

For our future work, there are two aspects that need further investigations: (1) We will apply DeepController to applications with other types of inputs, such as audios and text; (2) It is also important to investigate the influence of hyper-parameters (the probability adjusting factor ϵ of AS2, and the probability adjusting factor δ of AMS2) on the performance of DeepController.

6 Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61872039 and the Fundamental Research Funds for the Central Universities under Grant No. FRF-GF-19-019B.

References

[1] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated white-box testing of deep learning systems," in *Proceedings of SOSP'17*, 2017, pp. 1–18.

[2] H. Liang, X. Pei, X. Jia, W. Shen, and J. Zhang, "Fuzzing: State of the art," *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1199–1218, 2018.

[3] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 1–36, 2020.

[4] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "Deephunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proceedings of ISSTA'19*, 2019, pp. 146–157.

[5] H. B. Braiek and F. Khomh, "Deepevolution: A search-based testing approach for deep neural networks," in *Proceedings of ICSME'19*, 2019, pp. 454–458.

[6] S. Demir, H. F. Eniser, and A. Sen, "Deepsmartfuzzer: Reward guided test generation for deep learning," *arXiv:1911.10621*, 2019.

[7] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of ICSE'18*, 2018, pp. 303–314.

[8] H. Yang, F. Chen, and S. Aliyu, "Modern software cybernetics: New trends," *Journal of Systems and Software*, vol. 124, pp. 169–186, 2017.

[9] A. Odena, C. Olsson, D. Andersen, and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," in *Proceedings of ICML'19*, 2019, pp. 4901–4911.

[10] X. Xie, H. Chen, Y. Li, L. Ma, Y. Liu, and J. Zhao, "Coverage-guided fuzzing for feedforward neural networks," in *Proceedings of ASE'19*, 2019, pp. 1162–1165.

[11] P. Zhang, B. Ren, H. Dong, and Q. Dai, "Cagfuzz:coverage-guided adversarial generative fuzzing testing for image-based deep learning systems," *IEEE Transactions on Software Engineering*, 2021. DOI: 10.1109/TSE.2021.3124006.

[12] P. Zhang, Q. Dai, and S. Ji, "Condition-guided adversarial generative testing for deep learning systems," in *Proceedings of AITest'19*, 2019, pp. 71–72.

[13] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "Dlfuzz: Differential fuzzing testing of deep learning systems," in *Proceedings of ESEC/FSE'18*, 2018, pp. 739–743.

[14] S. Lee, S. Cha, D. Lee, and H. Oh, "Effective white-box testing of deep neural networks with adaptive neuron-selection strategy," in *Proceedings of ISSTA'20*, 2020, pp. 165–176.

[15] X. Du, X. Xie, Y. Li, L. Ma, J. Zhao, and Y. Liu, "Deepcruiser: Automated guided testing for stateful deep learning systems," *arXiv:1812.05339*, 2018.

[16] A. Rios, "Fuzze: Fuzzy fairness evaluation of offensive language classifiers on african-american english," in *Proceedings of AAAI'20*, vol. 34, no. 01, 2020, pp. 881–889.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009, TR-2009.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of ICLR'15*, 2015, pp. 134–146.

[21] L. St, S. Wold *et al.*, "Analysis of variance (anova)," *Chemometrics and intelligent laboratory systems*, vol. 6, no. 4, pp. 259–272, 1989.

WasmFuzzer: A Fuzzer for WebAssembly Virtual Machines

Bo Jiang, Zichao Li, Yuhe Huang
State Key Laboratory of Software Development Environment
School of Computer Science and Engineering
Beihang University
Beijing, China
{jiangbo, lizichao, yhhuang}@buaa.edu.cn

Zhenyu Zhang
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
zhangzy@ios.ac.cn

W.K. Chan
Department of Computer Science
City University of Hong Kong
Hong Kong
wkchan@cityu.edu.hk

***Abstract*—WebAssembly is a fast, safe, and portable low-level language suitable for diverse application scenarios. And The WebAssembly virtual machines are widely used by Web browsers or Blockchain platforms as execution engine. When there is a bug in the implementation of the Wasm virtual machine, the execution of WebAssembly may lead to errors or vulnerability in the application. Due to the grammar checks by WASM VMs, fuzzing at the binary level is ineffective to expose the bugs because most inputs cannot reach the deep logic within the WASM VM. In this work, we propose WasmFuzzer, a bytecode level fuzzing tool for WASM VMs. WasmFuzzer proposes to generate initial seeds for Fuzzing at the Wasm bytecode level and it also designs a systematic set of mutation operators for Wasm bytecode. Furthermore, WasmFuzzer proposes an adaptive mutation strategy to search for the best mutation operators for different fuzzing targets. Our evaluation on 3 real-life Wasm VMs shows that WasmFuzzer can significantly outperform AFL in terms of both code coverage and unique crash.**

Keywords-fuzzing; WebAssembly; Virtual Machine

I. INTRODUCTION

In order to improve the performance of Web applications, a number of companies and organizations have designed and implemented a new low-level language that can be executed across platforms, called WebAssembly [1].

WebAssembly was born in Web technology, and many browsers, including Chrome, have provided compatibility, allowing WebAssembly code files to be embedded in Web pages. WebAssembly modules will be able to call into and out of the JavaScript context and access browser functionality through the same Web APIs accessible from JavaScript. WebAssembly has some great features. First, it is a refined target language, with a significantly shorter code length than both scripting languages and many compiled native codes. As a result, it has a small footprint for deployment. Secondly, the instruction set of WebAssembly is designed to correspond directly to CPU instructions as much as possible. In some experiments, it runs more than 20 times faster than JavaScript and is more suitable for implementing more complex applications. Furthermore, since WebAssembly is a back-end language supported by LLVM [2], many source codes that support LLVM toolchains, including C [3], C++ [4], Rust [5], can be compiled into WebAssembly code, which allows much software originally implemented in traditional languages to generate WebAssembly code with the same functionality after adaptation, not only reducing the difficulty of program migration, but also allowing WebAssembly code to reuse existing library code. Because of these advantages, WebAssembly is now not only used as a technology for Web applications, but also integrated in blockchain platforms [6].

The WebAssembly code is executed within WebAssembly virtual machine [7]. The existing Wasm virtual machine implementations include WAVM [8], Wasmtime [9], Wasmer [10], etc. Virtual machines are the infrastructure that executes WebAssembly and should be implemented correctly, efficiently, and robustly. However, if there are errors in the implementation of the virtual machine, the execution of WebAssembly may lead to wrong results, or the program may exit abnormally. Some of these bugs can even lead to security vulnerabilities. For example, there are 7 CVEs reported for the Wasm VM called WAVM [8] in 2018. To avoid these situations, we can adopt fuzzing techniques [11] to identify errors in virtual machine implementations.

There are two major challenges faced with Wasm VM fuzzing. First, the Wasm VM often performs Wasm code validation before execution, which makes it hard to generate effective input to reach the deep logic within the VM. Although AFL, the mainstream fuzzing test software, can be used to test WebAssembly virtual machines written in C/C++, the test cases they generate are all binary data without considering Wasm bytecode grammar, which is hard to pass through the code validations commonly performed by the Wasm VMs. To solve this problem, we propose a Wasm bytecode level fuzzing framework that can both generate and mutate Wasm modules to test Wasm VMs. In particular, our proposed mutation operators can systematically mutate a Wasm module at different granularity. Second, there are many different implementations of

DOI reference number: 10.18293/SEKE2022-165

WASM VM and they have different code structures and bug patterns. A fixed mutation strategy is hard to accommodate the differences among those Wasm VMs to achieve the best fuzzing effectiveness. To solve this problem, we propose an adaptive mutation strategy that can dynamically update the probabilities of different mutation operators for a testing target.

The contributions of this work are as follows:

First, we propose a Wasm bytecode level fuzzing framework called WasmFuzzer for Wasm VMs, the tool can generate and mutate Wasm bytecode modules to reach the deep logic within the Wasm VMs.

Second, we propose an adaptive mutation strategy that can dynamically update the probabilities of different mutation operators. In this way, we can optimize the mutation operator configurations for a testing target.

Finally, we have systematically performed fuzzing on 3 real-life Wasm VMs with WasmFuzzer. Our evaluation results show that WasmFuzzer is more effective than AFL in terms of both code coverage and bug detection. And WasmFuzzer has detected 235 unique crashes within WAVM, WAMR, and EOS-VM.

The following sections are organized as follows. In section II, we will present the background knowledge on Wasm. In section III, we will discuss the design of WasmFuzzer in detail. In section IV, we present our fuzzing experiment with WasmFuzzer and AFL on 3 popular Wasm VM implementations and discuss the experiment results. Finally, we present related works and conclusion in section V and VI.

II. Background

In this section, we present background information on Wasm bytecode. In general, Wasm is a binary instruction format for a stack-based virtual machine. It is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications.

Wasm provides only four basic number types. These are integers and IEEE 754-2019 numbers, each in 32 and 64 bit width [1]. The computational model of WebAssembly is based on a stack machine. The instructions of Wasm fall into two main categories: simple instructions performing basic operations on data and control instructions altering control flow. The instructions are in turn organized into separate functions. A table in Wasm stores an array of untyped function references, which a program can call indirectly through a dynamic index into a table. WebAssembly adopts a linear memory structure, which is a contiguous, mutable array of raw bytes. A program can load and store values from/to a linear memory at any byte address. Finally, a WebAssembly binary takes the form of a module that contains definitions for functions, tables, linear memories, and global variables. In addition to definitions, modules can define initialization data for their memories or tables.

A. The Workflow of WasmFuzzer

The workflow of WasmFuzzer is shown in Figure 1. , which follows the general workflow of coverage-guided grey-box fuzzing. At first, WasmFuzzer will generate a set of Wasm files as seed inputs. Then it will enqueue these Wasm files and start the Wasm VM under fuzzing. Within the fuzzing loop, it will dequeue the first Wasm module and execute it against the Wasm VM. After execution, if the execution of the Wasm module leads to any new code coverage or new crashes, the module is considered a good candidate for mutation. And the WasmFuzzer will perform mutation on it to generate new Wasm modules, which are then enqueued for further fuzzing. Note that WasmFuzzer proposes several different mutation strategies to perform mutation. Then WasmFuzzer will further check the condition to stop the fuzzing process. If the fuzzing has reached the predefined time limit, the fuzzing will halt. Otherwise, it will continue the fuzzing loop the dequeue the next Wasm module for execution.

III. The Design of WasmFuzzer

A. The Generation of Wasm Bytecode

The input to the Wasm VM is the Wasm bytecode. To extensively fuzz the WebAssembly VM, WasmFuzzer proposes to generate valid Wasm bytecode for execution and mutation. Compared with binary input and mutation, the bytecode level inputs have a higher chance to reach deeper logic of the Wasm VM.

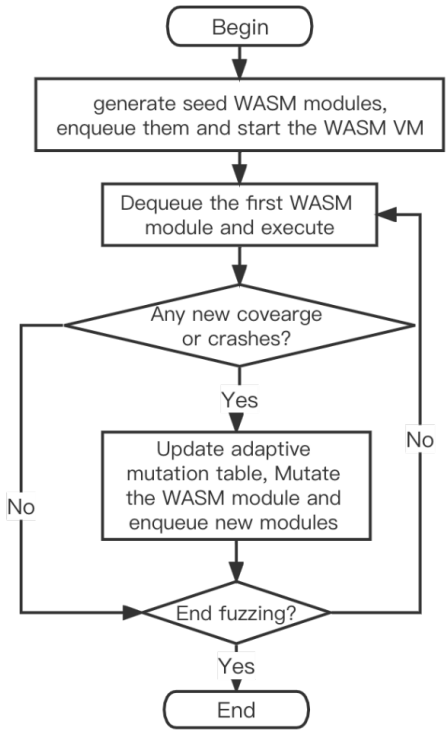


Figure 1. The Workflow of WasmFuzzer

According to the characteristics of the instruction, there are two main approaches to generating parameters: selecting parameters from the module and generating parameters from the domain of data type. Selecting a parameter from a module is used when the parameter of the instruction depends on the

internal state of the module. For example, the **global.set** instruction is to set a global variable at the top of the stack, and its parameter is the id of the global variable. Therefore, WasmFuzzer obtains the ids of all global variables from the global array in the module and selects one of them as the parameter of the instruction. Generating parameters from the domain of data type is used when the parameter is of certain data type. In such case, WasmFuzzer randomly returns a value within the domain of the data type.

WasmFuzzer extends the WebAssembly Binary Toolkit (WABT) to help generate different kinds of instructions. To be specific, it uses the internal functions of the WABT to generate different kinds of opcode, which are combined with the corresponding parameters to build different instructions. Finally, the instructions are further assembled into functions and modules as seed inputs.

B. Mutation Operator for Wasm Bytecode

Modules are the basic unit of deployment for WebAssembly. With an existing module, you can mutate it to generate new modules for fuzzing. To support feedback-directed fuzzing, we have systematically designed a set of mutation operators for Wasm modules.

1) Mutation operations

Mutation operations are divided into 2 types: mutation operations on instructions and other mutation functions. The mutation operations currently supported by WasmFuzzer are shown in TABLE I. .

TABLE I. LIST OF WASMFUZZER MUTATION OPERATIONS

Classification	***Mutation Operator***	***Description***
Mutation operations on Instructions	insertInstruction	Insert an instruction
	eraseInstruction	Delete an instruction
	moveInstruction	Move an instruction
	addFunction	Add an empty function
	eraseFunction	Delete a function
	swapFunction	Swap the positions of two functions
Other mutation operations	addGlobal	Add a global variable
	eraseGlobal	Delete a global variable
	swapGlobal	Swap the positions of two global variables
	addExport	Add an export entry
	eraseExport	Delete an export entry
	swapExport	Swap the positions of two export entries
	addType	Add a type
	addMemory	Add a block of storage space
	setStart	Set the start function
	eraseStart	Delete start function

The mutation operations on instructions are performed at the instruction level or at the function level. They randomly insert, delete, or change the instructions or functions to perform the mutation. The other mutation operations aim at changing the global variables, the export entries, the memory, or the start functions. To ensure the mutated WebAssembly code can pass through the validation [12] process of Wasm VM, we control the probability of different mutation operators such that the newly generated Wasm modules have a higher chance to be valid.

2) Adaptive Random Mutation Strategy

WasmFuzzer proposes an adaptive random mutation strategy to perform mutation. During the mutation step, each mutation operator has a probability to be selected. In general, our mutation strategy will reward the mutation operators leading to new code coverage or crash by dynamically increasing their probabilities. In this way, those more "promising" mutation operators have a higher chance to be selected.

To realize this, WasmFuzzer defines an adaptive mutation table, which is an array of function pointers of length 256. These function pointers may point to different mutation operators. The first 16 positions of this array are read-only areas and they correspond to the 16 mutation operators. In this way, WasmFuzzer ensures each mutation operator at least has a chance of 1/256 to be selected for performing mutation, which is not affected by the adaptive strategy.

TABLE II. ALGORITHM UPDATING ADAPTIVE MUTATION TABLE

Input	table: adaptive mutation table,
	func: pointer to the current mutation operator
Output	updated adaptive mutation table
1	#define NEW_PATH_REWARD 3
2	#define CRASH_REWARD 6
3	int increase = 0;
4	if (new paths found)
5	increase += NEW_PATH_REWARD;
6	if (new crash triggered)
7	increase += CRASH_REWARD;
8	for (int i = 0; i < increase; ++i) {
9	int num = randomBetween(16, 255);
10	table[num] = func;
11	}
12	return table;

The positions starting from 16 to the 255 can be both read and written, which is used for dynamically changing the selection probability of the mutation operators. At first, all positions in the table are initialized to various mutation operations with equal probability. The algorithm to update the adaptive mutation table is shown in Table II. During fuzzing, if the Wasm module obtained from a mutation operator called *M* leads to new code coverage or crash, WasmFuzzer will increase the selection probability of the mutation operator *M* (lines 3 to 7). Then, WasmFuzzer will generate a random number between 16 and 255 as index into the mutation table, and overwrite the position in the table corresponding to the index with the pointer of *M* (line 9 to line 10). In this way, the probability of those more effective mutation operators for a fuzzing target will increase gradually while those ineffective mutation operators for a target will decrease gradually. When testing multiple Wasm VMs, the

adaptive mutation strategy can automatically change the probability of each mutation operation to find the best mutation probability for each Wasm VM.

C. Test Oracle and Bug Report Generation

When the software under testing crashes or aborts during fuzzing, the system will send out signals such as SIGSEGV or SIGABT. WasmFuzer will capture these signals to report errors. Furthermore, WasmFuzzer also utilizes the AddressSanitizer [13] to detect memory-related software bugs such as use-after-free, buffer overflow, stack overflow, memory leaks, etc.

When WasmFuzzer has detected an error, it will generate bug reports to facilitate further debugging. The bug reports include two sections: the Wasm bytecode triggering a unique crash, and the Wasm bytecode triggering a unique hang. By "Unique", it means the execution of these Wasm bytecode leads to unique code path. Furthermore, we also measure the code coverage achieved during fuzzing as another metric.

IV. Evaluation

In this section, we evaluate WasmFuzzer by fuzzing 3 large-scale Wasm VMs.

A. Research Question

Based on the implementation of WasmFuzzer, this chapter mainly focuses on its test capability and test efficiency. Various performance metrics of WasmFuzzer and AFL were compared, including code coverage, number of unique crashes that could be found, and type of software problem, through comparative experiments under the same conditions.

B. Experiment Design

In our experiment, we compare WasmFuzzer with AFL to evaluate its fuzzing effectiveness.

1) Subjects

We have selected 3 real-life Wasm VM implementations to evaluate WasmFuzzer. These 3 Wasm VMs (WAVM, WAMR, and EOS VM) are written in C/C++, which is friendly for instrumentation and collecting code coverage. WAVM [8] is a popular WebAssembly virtual machine designed for non-browser applications. WebAssembly Micro Runtime [14] (WAMR for short) is a small WebAssembly virtual machine frequently used in embedded systems.

EOS-VM [15] is a WebAssembly virtual machine designed for blockchain applications. Since the command line interface provided by EOS-VM only supports the call of exported functions without parameters. To perform fuzzing, we modified the interface of EOS-VM to call the exported functions with parameters.

2) Experimental Setup

Our experiments were performed using a desktop with Intel(R) Core (TM) i7-6700 CPU @ 3.40 GHz and 16GB of memory. The operating system is Ubuntu 20.04 LTS. The version number of the AFL tool for comparison is 2.51b.

3) Instrumentation Procedure

To instrument the Wasm VMs for code coverage collection, we use Gcc compiler with code coverage profiling options enabled. To detect memory-related bugs, we also enabled the address sanitizer during compilation.

4) The experimental process

For each WebAssembly VM, we performed 8 hours of fuzzing using both WasmFuzzer and AFL. Then we use the afl-cov tool to analyze the code coverage achieved by each tool. We also manually analyzed the test cases leading to the crash or hang in the VMs to confirm the bug detected.

C. Results and Analysis

In this section, we present and compare the results of WasmFuzzer and AFL in terms of code coverage and unique crashes.

1) Code coverage

The code coverage results for the 3 Wasm VMS are shown in TABLE III. For WAVM, the code coverage of WasmFuzzer is 25.7% while the code coverage for AFL is 23.6%. For WAMR, the code coverage of WasmFuzzer is 25.9% while the code coverage for AFL is 22.7%. For EOS-VM, the code coverage of WasmFuzzer is 84.7% while the code coverage for AFL is 59.3%. We can see that WasmFuzzer consistently performs better than AFL in terms of code coverage at line level.

TABLE III. CODE COVERAGE RESULTS

Subjects	***Code Coverage***	
/	***WasmFuzzer***	***AFL***
WAVM	25.7%	23.6%
WAMR	25.9%	22.7%
EOS-VM	84.7%	59.3%

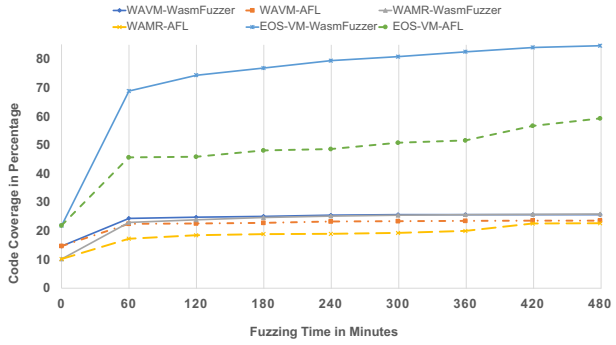


Figure 2. Code Coverage over Time

As shown in Figure 2, we also present the code coverage with respect to fuzzing time for WasmFuzzer and AFL on all 3 Wasm VMs. In this way, we want to understand the code coverage results of WasmFuzzer during the fuzzing process. We can see that for each subject VM, WasmFuzzer consistently performs better than AFL in terms of code coverage over time. And the advantage is more significant on EOS-VM than the other 2

Wasm VMs. Therefore, for different fuzzing time limit, WasmFuzzer can outperform AFL in terms of code coverage.

Based on the results above, we can conclude that WasmFuzzer can indeed achieve more code coverage than AFL if given the same fuzzing time.

2) Unique crashes

The main goal of fuzzing is to find bugs in the system. Therefore, we further present and compare the unique crashes detected by WasmFuzzer and AFL. The number of unique crashes for WasmFuzzer and AFL are shown in TABLE IV. We can see that WasmFuzzer consistently outperforms AFL on all three WebAssembly VMs. For WAVM, the difference between WasmFuzzer and AFL is small. But on WAMR and EOS-VM, the advantage of WasmFuzzer is significant.

In particular, for EOS-VM, the AFL fails to detect any error after 8 hours of fuzzing. We double-checked the code of EOS-VM, and we find that it performs strict code validation checks before executing the Wasm code. Most of the inputs generated by AFL are rejected during the code validation phase. As a result, AFL cannot detect the bugs hidden in the VM execution program logic. In contrast, WasmFuzzer can build and mutate valid Wasm modules, which makes it easier to test the execution logic of EOS-VM.

TABLE IV. NUMBER OF UNIQUE CRASHES

Subjects	***Unique Crash***	
/	***WasmFuzzer***	***AFL***
WAVM	56	55
WAMR	97	77
EOS-VM	82	0

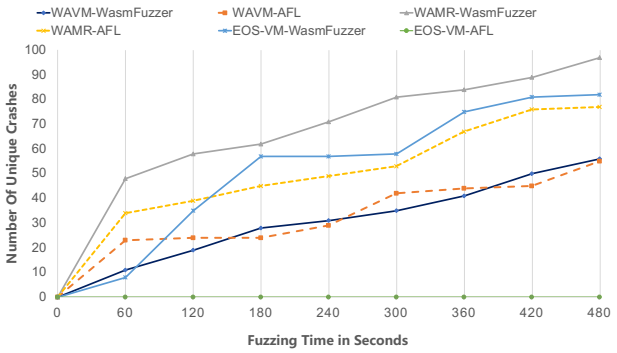


Figure 3. Unique Crashes over Time

The number of unique crashes over time for WasmFuzzer and AFL on the 3 Wasm VMs are shown in Figure 3. For WAMR and EOS-VM, WasmFuzzer consistently detected much more crashes than AFL over time. However, for WAVM, WasmFuzzer and AFL found almost the same number of unique crashes over time. A closer analysis on the crashes shows that WasmFuzzer and AFL can indeed detect different unique crashes. Therefore, when there are abundant resources during fuzzing, it is desirable to adopt both tools to perform fuzzing so they can complement each other. However, when the testing resource is limited, WasmFuzzer is preferred than AFL.

Based on the results above, we can conclude that WasmFuzzer can perform as good as or better than AFL in terms of unique crashes.

V. RELATED WORK

Park et al. designed a new test case mutation technique called aspect-preserving, and implemented a JavaScript fuzzing tool called DIE [16]. They believe that there are certain patterns in test cases that can trigger vulnerabilities. For the JavaScript language, the combination of some code structures and variable types is more likely to trigger vulnerabilities in the JavaScript execution engine. Therefore, DIE tends to retain these combinations when performing mutation.

Fuzzing tools can also be combined with neural network models to generate inputs that can trigger vulnerabilities more easily. Lee et al. developed a fuzzing tool based on neural network language model for JavaScript engines named Montage [17]. They train the model with the abstract syntax subtree converted from the JavaScript abstract syntax tree. In this way, the model can generate valid JavaScript code. With this approach, their tool has detected previously undiscovered software bugs in the JavaScript execution engine under fuzzing.

Zhong et al. designed and implemented a fuzzing tool called Squirrel for relational databases [18], whose input data is structured query language. Since the structured query language needs to meet certain grammatical rules, the proportion of input that can be executed by the database when directly mutating binary data is small. Therefore, they designed an intermediate representation capable of generating structured query language code and performed type-based mutation on the intermediate representation. In this way, the proportion of input that can be executed by the database is significantly increased.

Fuzzing tools are also effective to find functional implementation bugs in the software implementation. For example, Chen et al. implemented a fuzzing tool for differentially testing Java virtual machines [19]. The main idea is to use the same input to execute multiple Java virtual machines and compare the running results among them. If there is any difference in their results, one of these Java virtual machines must contain a bug. Engineers can further perform analysis and debugging based on the fuzzing results to find the position of the software error.

Ventuzelo proposes to use mainstream fuzzing tools to test WebAssembly virtual machines, and they integrated a fuzzing tool called WARF [20]. WARF can fuzz WebAssembly virtual machines and test them with binary data. WARF has found several bugs in the WebAssembly virtual machine implementation. WARF is implemented in Rust language and integrates three mainstream fuzzing tools, AFL++ [21], Honggfuzz [22] and libFuzzer [23].

VI. Conclusion

WebAssembly is a fast, safe, and portable low-level language suitable for diverse application scenarios. And The WebAssembly virtual machines are widely supported by Web browsers for building Web applications. When there is a bug in the implementation of the Wasm virtual machine, the execution of WebAssembly may lead to errors in its supporting application. Due to the code validation performed by WASM VMs, fuzzing at the binary level is ineffective to expose the bugs because most inputs cannot reach the deep logic within the WASM VM. In this work, we propose WasmFuzzer, a bytecode level fuzzing tool for WASM VMs. WasmFuzzer proposes to generate initial seeds for Fuzzing at the Wasm bytecode level and it also proposes a systematic set of mutation operators for Wasm bytecode. Furthermore, WasmFuzzer proposes an adaptive mutation strategy to search for the best mutation operators for different fuzzing targets. Our evaluation on 3 real-life Wasm VMs shows that WasmFuzzer can significantly outperform AFL in terms of both code coverage and unique crash.

For future work, we plan to explore new seed generation scheme and fuzzing input scheduling scheme to improve the effectiveness of the fuzzing tool. We will also perform fuzzing on other popular Wasm VMs to further evaluate the effectiveness of WasmFuzzer.

Acknowledgments

This research is supported in part by the National Key R&D Program of China under Grant 2019YFB2102400, NSFC (project no. 61772056), the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Innovative Technology Fund of HKSAR (project no. 9440226) and CityU MF_EXT (project no. 9678180). Zhenyu Zhang is the corresponding author.

References

[1] WebAssembly. https://webassembly.org/. Last access, 2022.

[2] The LLVM Compiler Infrastructure. https://llvm.org/. Last access, 2022.

[3] Ritchie D. M.. The Development of the C Language. *ACM Sigplan Notices 28.3*, 201-208, 1993.

[4] Stroustrup B.. The C++ programming language. Pearson Education India, India, 2000.

[5] Matsakis N. D.., Klock F. S.. The rust language. *ACM SIGAda Ada Letters 34.3*, 103-104, 2014.

[6] Wang S., Yuan Y., Wang X., et al. An overview of smart contract: architecture, applications, and future trends. *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[7] Sauntry D. M., Gilbert M.. Generating a compiled language program for an interpretive runtime environment. US, US6327702 B1. 2001.

[8] WAVM. https://wavm.github.io/. Last access, 2022.

[9] Wasmtime. https://wasmtime.dev/. Last access, 2022.

[10] Wasmer. https://wasmer.io/. Last access, 2022.

[11] Ammann P., Offutt J.. Introduction to software testing. *UK: Cambridge University Press*, 2016.

[12] Validation — WebAssembly 1.1 (Draft 2021-11-18). https://webassembly.github.io/spec/core/valid/index.html. Last access, 2021.

[13] AddressSanitizer. https://github.com/google/sanitizers/wiki/AddressSanitizer. Last access, 2019.

[14] WebAssembly Micro Runtime. https://github.com/bytecodealliance/wasm-micro-runtime. Last access, 2021.

[15] EOS VM - A Low-Latency, High Performance and Extensible WebAssembly Engine. https://github.com/EOSIO/eos-vm. Last access, 2019.

[16] Park S., Xu W., Yun I., et al. Fuzzing JavaScript Engines with Aspect-preserving Mutation. *2020 IEEE Symposium on Security and Privacy (SP)*, 1629-1642, 2020.

[17] Lee S., Han H. S., Cha S. K., et al. Montage: A Neural Network Language Model-Guided JavaScript Fuzzer. *20th USENIX Security Symposium (USENIX Security 2020)*. 2020.

[18] Zhong R., Chen Y., Hu H., et al. SQUIRREL: Testing Database Management Systems with Language Validity and Coverage Feedback. https://arxiv.org/abs/2006.02398, 2020.

[19] Chen Y., Su T., Sun C., et al. Coverage-directed differential testing of JVM implementations. *In Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2016: 85-99.

[20] WARF - WebAssembly Runtimes Fuzzing project. https://github.com/pventuzelo/wasm_runtimes_fuzzing. Last access, 2022.

[21] The AFL++ fuzzing framework | AFLplusplus. https://aflplus.plus/. Last access, 2021.

[22] Honggfuzz | honggfuzz. https://honggfuzz.dev/. Last access, 2021.

[23] libFuzzer – a library for coverage-guided fuzz testing. https://llvm.org/docs/LibFuzzer.html. Last access, 2022.

Multi-Frames Temporal Abnormal Clues Learning Method for Face Anti-Spoofing

Heng Cong*, Rongyu Zhang, Jiarong He, Jin Gao
Interactive Entertainment Group of Netease Inc, Guangzhou, China
{congheng, zhangrongyu, gzhejiarong, jgao}@corp.netease.com

***Abstract*—Face anti-spoofing researches are widely used in face recognition and has received more attention from industry and academics. In this paper, we propose the EulerNet, a new temporal feature fusion network in which the differential filter and residual pyramid are used to extract and amplify abnormal clues from continuous frames, respectively. A lightweight sample labeling method based on face landmarks is designed to label large-scale samples at a lower cost and has better results than other methods such as 3D camera. Finally, we collect 30,000 live and spoofing samples using various mobile ends to create a dataset that replicates various forms of attacks in a real-world setting. Extensive experiments on public OULU-NPU show that our algorithm is superior to the state of art and our solution has already been deployed in real-world systems servicing millions of users.**

I. Introduction

Face anti-spoofing (FAS) plays an important role in interactive AI systems and is also a challenging task without specific hardware equipped in the industry. The existing methods based on multi-modal information (e.g. infrared light, structured light, and light field) are widely used in scenarios, where there are specialized hardware devices. Although these methods [1]–[3] perform well in classification, they cannot be used on mobile devices on a broad scale.

Initially, the traditional manual feature is used to recognize face spoofing [4]–[6]. With the development of convolution neural network (CNN), classification techniques are excelled in the FAS task [7]–[9]. Recently, the application of face depth information has further improved the FAS effectiveness. The previous method performed excellently, but there are many shortcomings. In [10]–[13], single-frame information is used to estimate the face depth for video prediction, but the inter-frame information of the video is discarded. [14]–[17] achieved better effects by considering the inter-frame information, whereas the feature of face abnormal clues is not be utilized. [18] label face depth with the 3D camera equipped to focus on face detail, which is difficult to promote. By contrast, [19]–[21] convert 2D faces to 3D faces using PRNet, which has a lower label cost but a higher error rate. Especially in the prediction of large-angle side faces, [19]–[21] is not effective. The binary mask label is obtained by the middle part of the face, in [10], [22]–[25], where live and attack is marked as 1 and 0, respectively. However, [10], [22]–[25] ignores the face depth and edge texture information, which degrades the effect of the model. Moreover, many datasets such as NUAA [26], CASIA-MFSD [27], Replay-Attack [28], MSU-USSA [29], and OULU-NPU [30] released several FAS data that are collected in the laboratory state. However, because of the uncertainty of lighting, scenarios, device, etc., the samples in the laboratory varies greatly from the samples in the real world.

To address the above problems, we propose a temporal feature fusion network named EulerNet. The eulerian video magnification is introduced to extract temporal clues and fuse the inter-frame feature from the video. Residual Pyramid is designed in EulerNet to fuse features of different resolutions. We propose a feature-compressed attention module (FCAM) to process features in the temporal dimension. To balance the cost and accuracy, we propose a lightweight face labeling method based on face landmarks, which is faster than [18] and more accurate than [19]–[21]. The face position map obtained by the proposed labeling method is easier to model at fine-grain than depth map. Compared with binary mask, more gradient information of the face edges is retained by face position map. Finally, a large and comprehensive FAS dataset is built to simulate the real usage background under the industry environment. According to the real proportion of cell phone models and application scenarios in real-world, we collect live and attack by different types of mobile ends in various scenarios for our dataset.

In summary, the main contributions of this work as follows:

- We propose a novel network architecture to extract abnormal clues and process multi-frame temporal information. The feature-compressed attention module (FCAM) and Residual Pyramid are designed to improve the computational efficiency, focus on prominent face information, and amplify the weak signal.
- A lightweight face labeling method based on face landmarks is presented to balance the labeling cost and accuracy. Compared with binary mask and depth map, face position map is accurate and fast.
- We build a large and comprehensive dataset, in which live and attack samples are collected by different types of mobile ends in various scenarios.
- The effectiveness, performance, and advantages of the proposed method are numerically and experimentally verified.

* denotes corresponding author

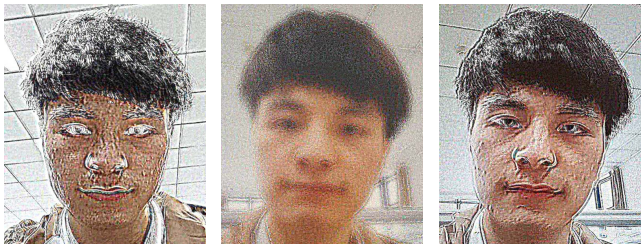
DOI reference number: 10.18293/SEKE2022-076

II. The Proposed Method

In this section, we first introduce a temporal feature fusion network based on eulerian magnification (EulerNet), including feature-compressed attention module (FCAM) and Residual Pyramid. Following EulerNet, a lightweight face labeling method based on face landmarks is described and the supervision method based on face location map is presented.

A. EulerNet

We propose a temporal feature fusion network based on eulerian magnification, namely EulerNet, to extract the abnormal clues from live and spoofing. By applying eulerian video magnification [31] to live and spoofing faces, the import clues for face anti-spoofing are discovered. As illustrated in Fig. 1, the result of print attack sample, lack a natural motion transition has obvious distinguishability with other attacks. Although replay attack and live sample have the same motion transition, the pattern characteristics are different, which provides reliable information for the network. Fig. 2 shows EulerNet architecture includes a Residual Pyramid and a series of feature-compressed attention modules (FCAM) and max pooling. Instead of processing the whole video, we extract a sequence (length 4 and frame interval 3) from the video as input to avoid the problems of excessive memory usage and high derivation time overhead. The attention structure was designed in FCAM to process the temporal features. Using differential infinite impulse response filtering, FCAM amplify the subtle changes in faces between different frames. In Fig. 2, the feature level is divided from 256×256 into 128×128, 64×64, and 32×32 in EulerNet, which is fused by Residual Pyramid. Different levels of features (128×128, 64×64, and 32×32) are sampled for residuals to obtain effective frequency for face anti-spoofing.



(a) Live (b) Print (c) Replay

Fig. 1: Results of Applying Eulerian Video Magnification

Feature-compressed Attention Module. In FCAM, the feature map channels are adjusted to 1 by the 3×3 convolution (Conv3×3), which synthesizes information from each channel. Inspired by the realization of speech signal [32], we transform the implementation of an infinite impulse response (IIR) filter on the image feature map to construct a differential infinite impulse response filter (DIIRF). By initializing a state matrix at the feature map size with 0, the output and the update for state in time sequence can be described as

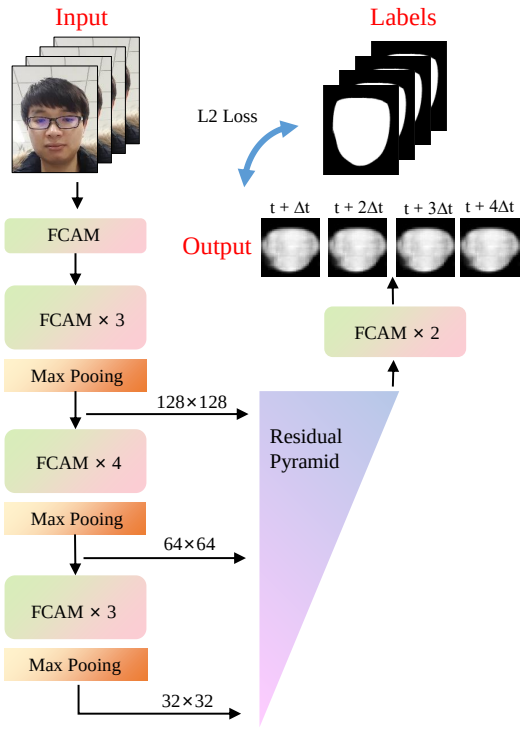


Fig. 2: The proposed EulerNet.

$$y[n] = b_0 x[n] + h_1[n-1] \tag{1}$$

$$h_1[n] = b_1 x[n] + h_2[n-1] - a_1 y[n] \tag{2}$$

$$h_2[n] = b_2 x[n] - a_2 y[n] \tag{3}$$

where $y[n]$ and $x[n]$ present output and input at nth timestamp, respectively. h_i is the state matrix with 0. b_i and a_j is the training parameters of the filter layer, $i \in [0,\ 2]$, $j \in [1,\ 2]$. The sequential feature is utilized to filter video signals in DIIRF and the effective frequency for face anti-spoofing is reserved. The sigmoid function as the gate control unit process the feature from DIIRF. Attentional structure is built by multiplying the feature map obtained by sigmoid back to the original input. FCAM architecture is shown in Fig. 3.

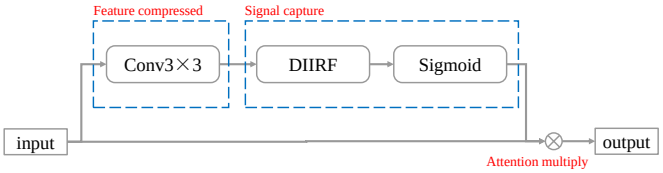


Fig. 3: Feature-compressed attention module.

Residual Pyramid. To amplify the weak signal for face anti-spoofing, we design Residual Pyramid to fuse feature. As shown in Fig. 2, 3-level outputs from backbone network are collected as the input of Residual Pyramid. In Residual Pyramid, the 32×32 ($F_{32\times32}$) and 64×64 ($F_{64\times64}$) features are upsampled to generate 64×64 ($F'_{64\times64}$) and 128×128

$F'_{128\times128}$) feature. Then, the $F'_{64\times64}$ and $F'_{128\times128}$ feature are subtracted by $F_{64\times64}$ and $F_{128\times128}$, respectively. After Conv3×3 extracting, the output $S_{64\times64}$ and $S_{128\times128}$ are obtained.

$$S_{128\times128} = Conv\left(F_{128\times128} - Upsample\left(F_{64\times64}\right)\right) \quad (4)$$

$$S_{64\times64} = Conv\left(F_{64\times64} - Upsample\left(F_{32\times32}\right)\right) \quad (5)$$

$$Ouput_{32\times32} = Concat(Donwsample(S_{128\times128}), Donwsample(S_{64\times64}), F_{32\times32}) \quad (6)$$

The Residual Pyramid can help the network directly learn useful information from features of different depths. In particular, upsampling and downsampling are introduced to calculate the residual between different resolutions.

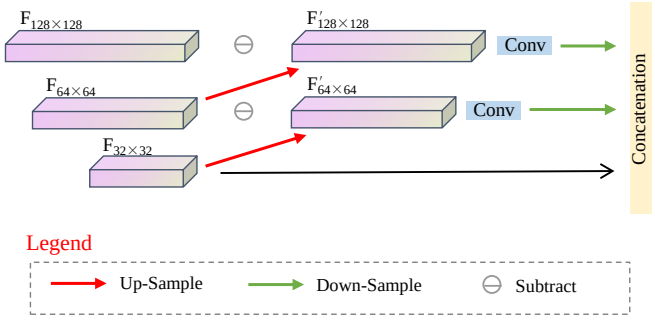


Fig. 4: Residual Pyramid for feature fusing of different depths.

B. A lightweight face labeling method based on face landmarks

Binary mask-based and Depth map-based labeling methods are effective, but both have drawbacks. Binary mask [22] does not distinguish between the face and background, and discards the depth information, which reduces the network efficiency. For depth map-based labeling methods, additional labeling resources are required. Compared with binary mask-based and Depth map-based labeling methods, we connect the key points of the face contour to form a closed face region, namely face position map. In particular, the pixel points within the face region are labeled as 1 and 0 on live face and spoofing, respectively. Face position map is used to reduce the labeling time, preserve the gradient information of the face edges, and improve accuracy. Fig. 5 shows the binary mask, depth map, and face position map.

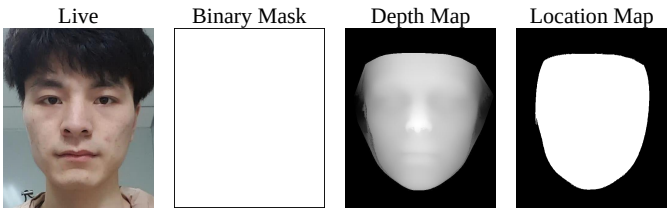


Fig. 5: Comparison of different supervision for FAS.

III. Dataset Collection

The difference of data distribution in different application scenarios determines the model performance in a specific scenario. To address this problem, we simulate the real-world scenario to establish the face anti-spoofing dataset, where each data sample is captured by the mobile end camera. In our dataset, the majority of mobile ends currently on the market are used for data collection. The spoofing type is divided into three parts: complex attack, certificate attack, and image dynamic generation attack. In particular, it is difficult to create and conduct 3D attacks in practical scenarios, so we mainly collect 2D attacks used in [33].

In our dataset, the complex attack includes various paper attacks, projection attacks, high-definition quality shots, and low-quality shots. Certificate attack is common in real-world scenarios and different from other attacks, which consists of ID card attacks, passport document attacks, etc. Image dynamic generation attack is an AI attack method, where dynamic images are generated from single images for face spoofing.

Our dataset outperforms previous public datasets [27], [28], [34] in three points: 1) Our dataset is the largest one that includes 30,000 live and spoofing videos (average duration to be 2s), collected from 10000 subjects, compared to 12,000 videos from 200 subjects in [18]. 2) As shown in Fig. 6, our dataset covers dozens of commonly used mobile ends. 3) The data distribution of our dataset matches the real-world mobile verification scenarios.

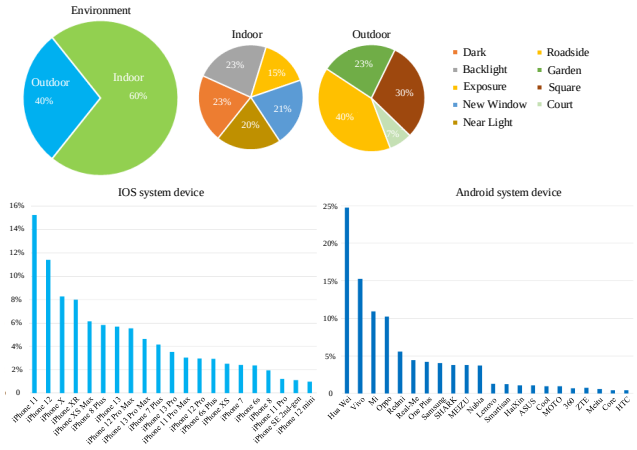


Fig. 6: Statistics of electronic devices and scenes of our dataset.

IV. Experiments

A. Implementation Details

To verify the effectiveness of our proposed network architecture and supervision method, we conduct experiments on the collected dataset, dividing the training, development, and test sets according to the ratio of 3:1:1. The learning rate is initialized as 1e-4 and the batchsize is set to 16 for a synergistic optimizer Ranger [35]. The input to the model are

sequences of 4 consecutive frames taken from the same video at intervals of 3. Thus we compute 64 images in one gradient update. The model is trained with Nvidia Tesla V100s. Pytorch is utilized as the backend for the network architecture. L2 loss is utilized to guide the network predict the location map.

Average Classification Error Rate (ACER) is the mean value of Attack Presentation Classification Error Rate (APCER) and Bona Fide Presentation Classification Error Rate (BPCER), given by

$$APCER = \frac{FP}{TN + FP}, \quad BPCER = \frac{FN}{TP + FN} \tag{7}$$

$$ACER = \frac{APCER + BPCER}{2} \tag{8}$$

In the next subsections, ACER is uniformly utilized for evaluation. For each video, we extracted multiple non-overlapping sequences, then calculated the live score individually and finally average scores for video.

B. Ablation Study

To investigate the behavior of EulerNet, we conducted several ablation studies. Table I shows the ablation study on our dataset. The baseline model is the proposed method using all improvements. Compare 1 (C1) and Compare 5 (C5) discuss the influence of different labels, including binary mask and depth map. In Compare 3 (C3), the Residual Pyramid is replaced by channels concatenation. Compare 4 (C4) removes the FCAM. Compare 2 (C2) discards all structure improvements and uses the common depth map for global comparison.

Effectiveness of FCAM and Residual Pyramid. After adding FCAM and d Residual Pyramid, ACER decreased by 0.34 percent and 0.38 percent, respectively, indicating that they were beneficial in FAS. The results suggest that extracting the effective frequency information of the input and fusing different resolution features together in a residual manner is meaningful for face anti-spoofing. Table I demonstrate that we must guide the network to mine as many anomalous patterns as possible in the attack video. The ACER of the proposed model is reduced by 0.7% concerning Compare 2 in Table I. We optimize the combination of models and labels to maximize the benefits and obtain excellent accuracy on our dataset.

Effectiveness of Face Location Map Supervision. The optimal model on the development set is utilized to check the test set. Our proposed model yields the best ACER, achieving 0.18% lower than the model supervised with depth map and 0.96% lower than the model supervised with binary mask. This indicates that the features with binary mask supervision are not robust and discriminative, which are learned from labeling all pixels of living images as 1. The practice of regression on face location map reduces the difficulty of pixel modeling task and makes the network focus more on the color texture of face region and edge gradient information.

As shown in the Fig. 7, we plot the training performance curves of different models. At the beginning, the curve without FCAM drops the fastest because the model does not extract frequency information. Although the early iteration is fast, the accuracy in the later stage is low, which suggests that it is significant to mine the specific frequency information. In the late training phase (epochs 45-50), the fluctuating curves are sorted by mean value size as C2>C1>C4>C3>C5>our method. Especially, the proposed method curve shows a smoother decreasing trend during training with less fluctuation.

TABLE I: Ablation study on our dataset.

Tag	Structure			ACER(%)↓	
	Label	FCAM	Residual Pyramid	Dev	Test
Compare 1	Binary Mask	√	√	3.95	2.84
Compare 2	Depth Map	×	×	3.62	2.57
Compare 3	Face Location Map	√	×	2.85	2.26
Compare 4	Face Location Map	×	√	3.13	2.22
Compare 5	Depth Map	√	√	2.74	2.06
Baseline	Face Location Map	√	√	**2.48**	**1.88**

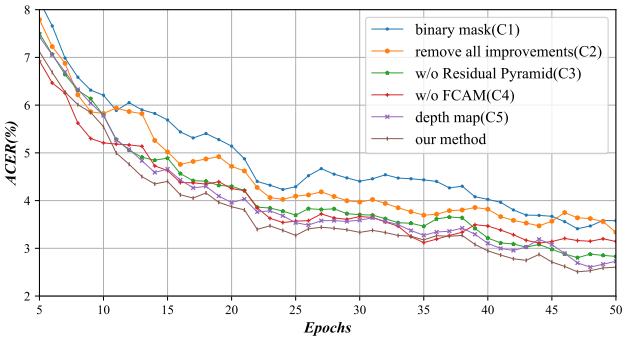


Fig. 7: Performance comparison curve on the development set in training process.

C. Visualization

The Grad-cam [38] is used to visualize the output of the neurons in the model to compare the changes brought by changing the structure or the supervision method. Fig. 8 details

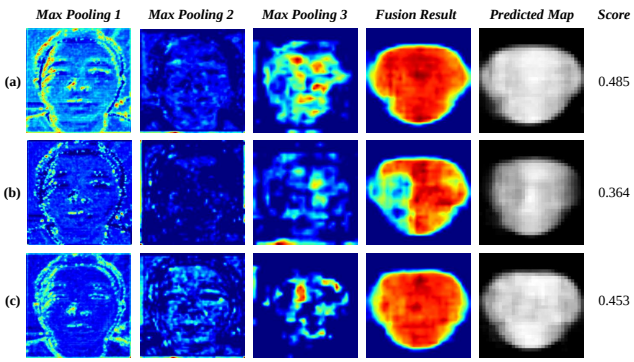


Fig. 8: Visualization study of neuron outputs at different depths: (a) our method, (b) w/o FCAM, (c) w/o Residual Pyramid. From left to right are the results with increasing downsampling multiplier as well as the multi-resolution fusion results and the final prediction maps. Their resolutions are 128×128, 64×64, 32×32, 32×32, 32×32 in order.

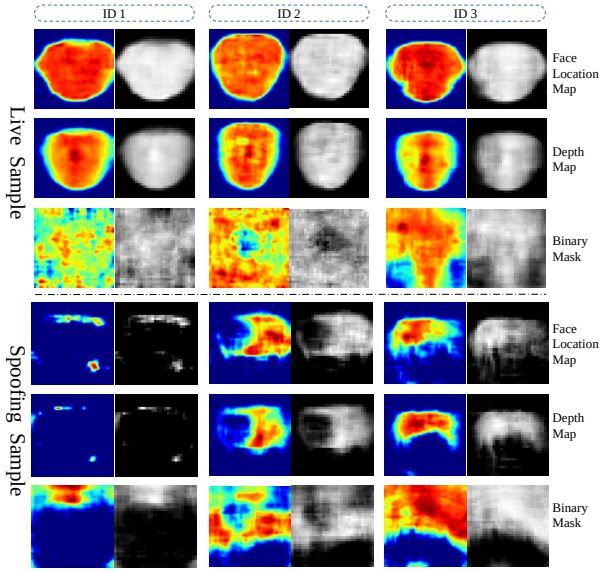


Fig. 9: Model output and heatmap visualization under different supervision methods. The same column indicates the results of the same face, marked with ID 1-3.

TABLE II: The results of intra testing on four protocols of OULU-NPU compared with deep learning methods

Prot.	Method	APCER(%)	BPCER(%)	ACER(%)
1	Disentangled [36]	1.7	0.8	1.3
	FAS-SGTD [14]	2.0	**0.0**	1.0
	DeepPixBiS [22]	0.8	**0.0**	**0.4**
	CDCN [37]	**0.4**	1.7	1.0
	EulerNet(Ours)	**0.4**	3.3	1.9
2	DeepPixBiS [22]	11.4	**0.6**	6.0
	Disentangled [36]	**1.1**	3.6	2.4
	FAS-SGTD [14]	2.5	1.3	1.9
	CDCN [37]	1.5	1.4	**1.5**
	EulerNet(Ours)	2.1	1.4	1.7
3	DeepPixBiS [22]	11.7±19.6	10.6±14.1	11.1±9.4
	FAS-SGTD [14]	3.2±2.0	2.2±1.4	2.7±0.6
	CDCN [37]	**2.4±1.3**	2.2±2.0	2.3±1.4
	Disentangled [36]	2.8±2.2	1.7±2.6	2.2±2.2
	EulerNet(Ours)	2.6±1.3	**1.6±0.8**	**2.1±0.5**
4	DeepPixBiS [22]	36.7±29.7	13.3±14.1	25.0±12.7
	CDCN [37]	4.6±4.6	9.2±8.0	6.9±2.9
	FAS-SGTD [14]	6.7±7.5	**3.3±4.1**	5.0±2.2
	Disentangled [36]	5.4±2.9	3.3±6.0	4.4±3.0
	EulerNet(Ours)	**1.8±1.9**	4.3±2.4	**3.1±0.9**

the effect of FCAM and Residual Pyramid on the features extracted by the network. Max Pooling 1-3 indicates the results using different downsampling multipliers. Fusion result is the multi-resolution fusion results. The final prediction are shown in Predicted Map. Scores are presented in the last column of Fig. 8. The higher the score on the far right, the higher the likelihood of being judged as live faces. The comparison between Max Pooling 1 and Max Pooling 2 shows that the visualization features are clearly distinct under different downsampling scales. The model with FCAM pays more attention to the parts where the action occurs, so there are higher activation values at pixels of the contour, eyes, and nose. Finally, the face score obtained by our method is 0.485, which is better than the model removed FCAM and Residual Pyramid. In addition, as the network deepened, the extracted features mined more abnormal clues and produced more accurate face location maps.

Fig. 9 shows the visualization results for the three different labels (face location map, binary map, and depth map) are studied in this paper. For the live face, the heat map supervised by the face localization map is sharper and finer at the edges of the face region. As shown in Fig. 9, the binary mask-based supervised method ignores the gradient information of face edges. Compared with the depth map, the prediction map based on the face location map has higher contrast in distinguishing faces and backgrounds. When the input is spoofing face, the predictions obtained by the face location map-based and depth map-based methods are similar and still bias the face region, which demonstrates that the network does not need to simulate depth information in practice. As shown in Fig. 9, the results of the binary mask have less distinct semantic features. The data distribution on the heat map is more uniform and scattered with the binary mask supervision. Compared with the above three supervision methods, the loss of generalization of the features extracted by the network without the drive of depth and location information.

D. Comparison on Public Dataset

We compare the proposed EulerNet with recent deep learning methods on OULU-NPU [30]. Protocol 1-3 respectively evaluate the generalization of the algorithm under previously unseen environmental conditions, attacks created with different printers or displays, and input camera variation. Protocol 4 considers all the above three factors. Generally, protocols 3 and 4 are more difficult than other protocols.

The Comparison results on OULU-NPU are shown in Table II. The best performance obtained by the proposed method in protocols 3 and 4 demonstrates that our method can maintain accuracy under complex conditions. Table II proves that the structures we have designed are capable of extracting abnormal clues, which has a stronger generalization in facing changes of the shooting device. The complexity of protocols 3 and 4 is similar to the realistic scenario where electronic products are changing rapidly, there are demonstrates the practicality of our proposed method.

V. Conclusion

In this paper, we propose a novel face anti-spoofing method, which effectively recognize the subtle differences between real face and spoofing in the video. The novel network architecture, namely EulerNet, is designed to fuse temporal information and extract abnormal clues. We propose a lightweight labeling method based on face landmarks to reduce the labeling cost and improve the labeling speed. Extensive experimental results on our datasets and public OULU-NPU validate the effectiveness of our method.

REFERENCES

[1] Y. Liu, A. Jourabloo, and X. Liu, "Learning deep models for face anti-spoofing: Binary or auxiliary supervision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 389–398.

[2] X. Xie, Y. Gao, W.-S. Zheng, J. Lai, and J. Zhu, "One-snapshot face anti-spoofing using a light field camera," in *Chinese Conference on Biometric Recognition*. Springer, 2017, pp. 108–117.

[3] D. Yi, Z. Lei, Z. Zhang, and S. Z. Li, "Face anti-spoofing: Multi-spectral approach," in *Handbook of Biometric Anti-Spoofing*. Springer, 2014, pp. 83–102.

[4] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face anti-spoofing based on color texture analysis," in *2015 IEEE international conference on image processing (ICIP)*. IEEE, 2015, pp. 2636–2640.

[5] T. A. Siddiqui, S. Bharadwaj, T. I. Dhamecha, A. Agarwal, M. Vatsa, R. Singh, and N. Ratha, "Face anti-spoofing with multifeature videolet aggregation," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1035–1040.

[6] X. Li, J. Komulainen, G. Zhao, P.-C. Yuen, and M. Pietikäinen, "Generalized face anti-spoofing by detecting pulse from face videos," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 4244–4249.

[7] J. Stehouwer, A. Jourabloo, Y. Liu, and X. Liu, "Noise modeling, synthesis and classification for generic object anti-spoofing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7294–7303.

[8] O. Lucena, A. Junior, V. Moia, R. Souza, E. Valle, and R. Lotufo, "Transfer learning using convolutional neural networks for face anti-spoofing," in *International conference image analysis and recognition*. Springer, 2017, pp. 27–34.

[9] Y. A. U. Rehman, L. M. Po, and M. Liu, "Deep learning for face anti-spoofing: An end-to-end approach," in *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. IEEE, 2017, pp. 195–200.

[10] Z. Yu, Y. Qin, X. Xu, C. Zhao, Z. Wang, Z. Lei, and G. Zhao, "Auto-fas: Searching lightweight networks for face anti-spoofing," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 996–1000.

[11] Z. Yu, J. Wan, Y. Qin, X. Li, S. Z. Li, and G. Zhao, "Nas-fas: Static-dynamic central difference network search for face anti-spoofing," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 9, pp. 3005–3023, 2020.

[12] Y. Qin, W. Zhang, J. Shi, Z. Wang, and L. Yan, "One-class adaptation face anti-spoofing with loss function search," *Neurocomputing*, vol. 417, pp. 384–395, 2020.

[13] Y. Qin, Z. Yu, L. Yan, Z. Wang, C. Zhao, and Z. Lei, "Meta-teacher for face anti-spoofing," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[14] Z. Wang, Z. Yu, C. Zhao, X. Zhu, Y. Qin, Q. Zhou, F. Zhou, and Z. Lei, "Deep spatial gradient and temporal depth learning for face anti-spoofing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5042–5051.

[15] M. Asim, Z. Ming, and M. Y. Javed, "Cnn based spatio-temporal feature extraction for face anti-spoofing," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 2017, pp. 234–238.

[16] Z. Yu, X. Li, P. Wang, and G. Zhao, "Transrppg: Remote photoplethysmography transformer for 3d mask face presentation attack detection," *IEEE Signal Processing Letters*, vol. 28, pp. 1290–1294, 2021.

[17] X. Yang, W. Luo, L. Bao, Y. Gao, D. Gong, S. Zheng, Z. Li, and W. Liu, "Face anti-spoofing: Model matters, so does data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3507–3516.

[18] Y. Liu, Y. Tai, J. Li, S. Ding, C. Wang, F. Huang, D. Li, W. Qi, and R. Ji, "Aurora guard: Real-time face anti-spoofing via light reflection," *arXiv preprint arXiv:1902.10311*, 2019.

[19] Z. Yu, X. Li, X. Niu, J. Shi, and G. Zhao, "Face anti-spoofing with human material perception," in *European Conference on Computer Vision*. Springer, 2020, pp. 557–575.

[20] T. Kim, Y. Kim, I. Kim, and D. Kim, "Basn: Enriching feature representation using bipartite auxiliary supervisions for face anti-spoofing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[21] Z. Wang, C. Zhao, Y. Qin, Q. Zhou, G. Qi, J. Wan, and Z. Lei, "Exploiting temporal and depth information for multi-frame face anti-spoofing," *arXiv preprint arXiv:1811.05118*, 2018.

[22] A. George and S. Marcel, "Deep pixel-wise binary supervision for face presentation attack detection," in *2019 International Conference on Biometrics (ICB)*. IEEE, 2019, pp. 1–8.

[23] M. S. Hossain, L. Rupty, K. Roy, M. Hasan, S. Sengupta, and N. Mohammed, "A-deeppixbis: Attentional angular margin for face anti-spoofing," in *2020 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2020, pp. 1–8.

[24] Y. Ma, L. Wu, Z. Li *et al.*, "A novel face presentation attack detection scheme based on multi-regional convolutional neural networks," *Pattern Recognition Letters*, vol. 131, pp. 261–267, 2020.

[25] W. Sun, Y. Song, H. Zhao, and Z. Jin, "A face spoofing detection method based on domain adaptation and lossless size adaptation," *IEEE access*, vol. 8, pp. 66 553–66 563, 2020.

[26] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," in *European Conference on Computer Vision*. Springer, 2010, pp. 504–517.

[27] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li, "A face anti-spoofing database with diverse attacks," in *2012 5th IAPR international conference on Biometrics (ICB)*. IEEE, 2012, pp. 26–31.

[28] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *2012 BIOSIG-proceedings of the international conference of biometrics special interest group (BIOSIG)*. IEEE, 2012, pp. 1–7.

[29] K. Patel, H. Han, and A. K. Jain, "Secure face unlock: Spoof detection on smartphones," *IEEE transactions on information forensics and security*, vol. 11, no. 10, pp. 2268–2283, 2016.

[30] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid, "Oulu-npu: A mobile face presentation attack database with real-world variations," in *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*. IEEE, 2017, pp. 612–618.

[31] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman, "Eulerian video magnification for revealing subtle changes in the world," *ACM transactions on graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[32] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable iir filters for machine learning applications," in *Proc. Int. Conf. Digital Audio Effects (eDAFx-20)*, 2020, pp. 297–303.

[33] L. Li and X. Feng, "Face anti-spoofing via deep local binary pattern," in *Deep Learning in Object Detection and Recognition*. Springer, 2019, pp. 91–111.

[34] P. P. Chan, W. Liu, D. Chen, D. S. Yeung, F. Zhang, X. Wang, and C.-C. Hsu, "Face liveness detection using a flash against 2d spoofing attack," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 2, pp. 521–534, 2017.

[35] L. Wright, "Ranger - a synergistic optimizer." https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2019.

[36] K.-Y. Zhang, T. Yao, J. Zhang, Y. Tai, S. Ding, J. Li, F. Huang, H. Song, and L. Ma, "Face anti-spoofing via disentangled representation learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 641–657.

[37] Z. Yu, C. Zhao, Z. Wang, Y. Qin, Z. Su, X. Li, F. Zhou, and G. Zhao, "Searching central difference convolutional networks for face anti-spoofing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5295–5305.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

Paying Attention to the Insider Threat

Eduardo Lopez
Information Systems, DeGroote School of Business
McMaster University
Hamilton, ON, Canada
lopeze1@mcmaster.ca

Kamran Sartipi
Department of Computer Science
East Carolina University
Greenville, NC, USA
sartipik16@ecu.edu

***Abstract*—The misuse of information systems by internal actors – the insider threat – is an ever-growing concern in organizations of all types. The timely detection of an insider threat is as important as it is difficult. Analyzing user behaviors recorded in electronic logs require significant computing resources and the capability to find and interpret complex patterns in temporal sequences that may contain irrelevant, temporary or novel elements. In this paper we use an attention-based architecture derived from BERT (Bidirectional Encoder Representations from Transformers) for the creation, storage and updating of an always-current, holistic user behavior model that enables real-time insider threat detection through anomaly detection and user behavior prediction techniques. A case study with a very large transaction system is provided.**

Keywords: insider threat, transformers, BERT, anomaly detection, cybersecurity[1]

I. Introduction

Detecting an on-going insider threat is a significant challenge. Although the actions by every user are regularly recorded in electronic files (i.e., logs), those logs can be obscured by a very large number of unrelated events. Information in electronic logs is usually unstructured, stored in very large text files that require specialized tools to be analyzed. User behaviors are captured in sequences of events that can be mined for abnormal patterns. However, their ever-evolving characteristics and dependence on a variety of contextual parameters (e.g., the time of use or location of computer) pose remarkable challenges for advanced analytic applications. For the purpose of this work, we focus on a set of technologies known as *Transformers* [13] that have dramatically improved Artificial Intelligence (AI) applications in Natural Language Processing (NLP). They are encoder/decoder architectures that implement the concept of *Attention* [3].

The preceding decade has witnessed remarkable advances in Artificial Intelligence (AI). Deep learning in particular has consistently delivered results across myriad disciplines, sometimes surpassing human-level benchmarks [1]. A deep learning architecture is a multi-layer stack of neural network units where usually most of them are subject to learning and that may include non-linear input-to-output mappings [8]. Each layer in the network incrementally learns about the structure of data from its preceding layers, becoming very good function approximators that can find and learn very complex patterns in the data. There are several deep learning architectures that can be applied to specific applications and data types.

Deep learning has been proven effective across a wide range of disciplines, with cybersecurity being the focus of this work. The protection of information assets against malicious threats is a pivotal element for an increasingly technified society where information systems enable processes for many organizations. Arguably, one of the most interesting challenges pertains to the phenomena known as the insider threat. It can be defined as current (or former) users – or somebody impersonating them – that intentionally misuse access privileges negatively impacting the confidentiality, integrity or availability of information or information systems [4]. As an anomaly detection exercise, deep learning is an effective tool to find such patterns. User behavior is non-linear, complex and difficult to ascertain unless a strong function approximator such as a deep learning network can be used.

In this study we train a user behavior model as the baseline to perform anomaly detection and behavior prediction for insider threat detection. In particular, this paper contributes to the cybersecurity literature by proposing a Transformer-based architecture that uses self-attention for modeling user behaviors. The model created can be used effectively in transfer learning tasks.

This paper is organized as follows: Section II presents the relevant research to our work. Section III articulates the architecture of Transformers. Section IV presenst the proposed approach and Section V elaborates on the case study we conducted. Section VI draws conclusion to our discussion.

II. Related Work

The detection of the insider threat with machine learning can be formulated in multiple ways. Lopez and Sartipi [10] articulated the analysis as a supervised-learning classification, where the existence of a known pattern or signature is used by a logistic regression classifier to determine if the user behavior indicates information systems misuse. Liu et al [9] present an example for the use of autoencoders in insider threat detection. Paul and Mishra [12] propose an LSTM-based approach to threat detection. LSTM architecture' effectiveness in insider threat detection has also been demonstrated by Lopez and Sartipi [11], and by Paul and Mishra [12]. More advanced

[1]DOI reference number: 10.18293/SEKE2022-059

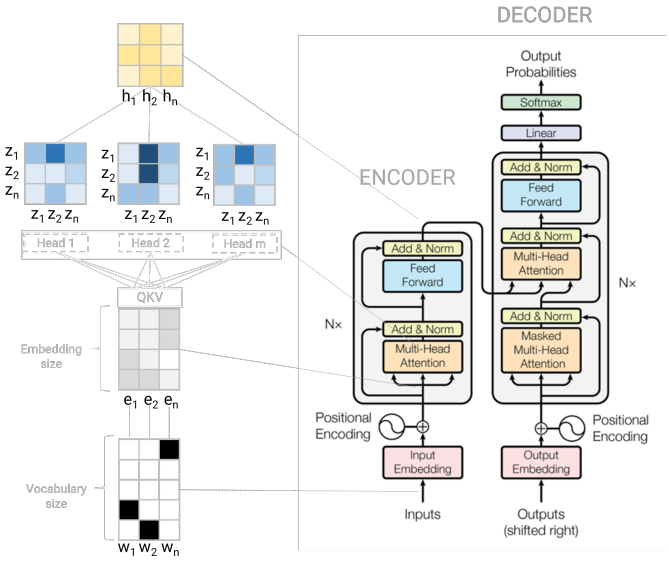


Fig. 1. Transformer architecture adapted from [13] .

architectures leveraging LSTM include stacked models, pre-initialized LSTM and bi-directional LSTM which have produced better results than single LSTM layers [6]. The concept of attention and in particular the Transformers proposed in the seminal work by Vaswani et al. [13]. From that moment on, the rate of new advances in Natural Language Processing has been unrelenting. Interestingly enough, the use of Transformers in cybersecurity applications has been rather limited. To our knowledge, no work has been done in applying these concepts to insider threat detection.

III. Transformer

The concept of *Attention* first appeared in 2015 [3] but it was until the Transformers concepts were proposed in 2017 that a new leap in performance was attained [13]. In its original form, a Transformer follows an encoder-decoder architecture as depicted in Figure 1. The original implementation uses six stacked encoders and six stacked decoders; however, since then multiple evolutions have taken place with different values and parameters. As it is the case with most deep learning architectures, the learning model uses successive representations of the original data, obtaining features at a higher abstraction level.

The input to a transformer model is usually raw data in the form of a sequence. Figure 1 represents this as a one-hot encoded table from words w_1 to w_n. The first step in the encoder is to transform the input into meaningful dense-vector representations. In contrast to recurrent neural networks, all the words in the sequence are fed at the same time to the encoder, so the word placement in the sequence is not explicitly captured. To add this information, a positional encoding takes place to calculate the words' positional embeddings. The encoder usually has multiple, parallel self-attention heads used during training. Each head can be considered a representation subspace where different attributes of the embeddings are calculated. In the case of NLP, a useful intuition is to consider each head as a representation of a different words' attributes, i.e., one may represent grammar and the other one may represent syntax or gender. The original Transformer uses eight heads with each one randomly initialized at training time.

Self-attention is the next step in the encoder which is considered fundamental to the concept of transformers. Reverting back to NLP for intuition: let us supposed we have the sentence *"Transformers are great for long sequences because **they** use attention"*. Any learning model needs to clearly determine if the word *they* refers to "Transformers" or to "long sequences". A way to formulate and quantify this dependency is by calculating for each word a vector to represent the contribution of itself and every other word in the sentence. Figure 1 depicts the self-attention as *m* square matrices of length *n*, where *m* is the number of attention heads in the architecture. Equation 1 shows the different operations performed.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Q, *K* and *V* are three abstractions for Query, Key and Value, which are calculated by multiplying the word embeddings with weigh matrices W^Q, W^K and W^V learned during training. The dot product of *Q* and *K* is scaled down by the square root of the dimension of the *K* matrix. A softmax transformation is then applied (producing a matrix with values from 0 to 1) and then multiplied by the V matrix to produce the attention matrix.

There are multiple transformer designs based on the original work by Vaswani [13]. For reasons that will be explained in section IV, we proceed to elaborate on the architecture known as BERT. The term stands for Bi-directional Encoder Representation from Transformers [5], and it uses only the encoder part of the transformer architecture. A fundamental characteristic of BERT is that it learns the underlying patterns of the data considering both the left and right context of the sequences used. BERT uses two unsupervised learning objectives: a Masked Language Model (MLM) and Next Sentence Prediction (NSP). These two tasks are executed over a large text corpus during a stage known as *pre-training*. Once the parameters have been found, the model is ready for prediction or classification tasks during a *fine-tuning* stage. Pre-training BERT can be conceptualized as building the underlying knowledge about the dataset. This knowledge can be then applied to other tasks with a smaller dataset, i.e., an example of *transfer learning*. While pre-training may be a lengthy task, fine-tuning is very efficient. Pre-trained BERT models are publicly available for NLP tasks. BERT large uses 24 encoders stacked, representing each word with 1024 dimensions and 16 self-attention heads. The total number of parameters learned for BERT large is 340M.

IV. Approach

Dataset. We use the Los Alamos cybersecurity events dataset that is publicly available for research [7]. It includes multiple files from different information systems containing authentication events, programs started or finished, Domain

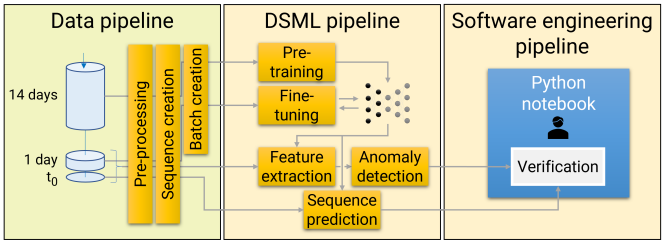


Fig. 2. Architecture for insider threat detection.

Name Service (DNS) calls and network flows. The dataset is the result of 58 days of continuous monitoring of more than 12,000 users and 17,000 computers. There are a total of 1.6 billion records contained in 17GB of data. In addition to this, there are 728 authentication records capturing the events from a Red Team, which are individuals purposely performing tasks that are typical of insider threats.

Architecture. To describe the architecture selected for the experiments, we use industry best practices, categorizing three groups of processes: i) Data engineering pipeline; ii) Data Science and Machine Learning (DSML) pipeline, and; iii) Software engineering pipeline [2]. In the data pipeline we transform the raw data into information that can be entered in the DSML model, with its output provided through a user interface so the user can verify the results.

We select BERT as the foundational architecture for the DSML pipeline. As was explained in Section III, BERT is an encoder that uses left and right contexts from sequences. Upon pre-training, BERT effectively creates a model (equivalent to a Masked Language Model in the original BERT implementation) that can be used in downstream tasks such as prediction or classification. In the scenario explored in this study, BERT produces a user behavior model that can be leveraged for the detection of the insider threat. In order to provide BERT with the required data, the data engineering pipeline pre-processes the data and crafts the sequences that will be used in the DSML model. Figure 2 depicts the complete architecture.

There are three distinct time windows used for the insider threat detection. The first one is the historical data used for pre-training. Based on domain knowledge, 14 days provides sufficient data to capture repetitive user patterns. Keeping the time window short, the pre-training time can be optimized so the learning process and can be completed in a timely fashion. Once the model has been pre-trained, a daily fine-tuning can take place. By using this strategy, the user behavior model is kept current with novel patterns performed by users. Fine tuning with 1-day data can be done overnight. The final time window used is one second. This means that the system will use the information collected within the log in real-time, and performs the analysis using the user behavior model.

The DSML component is where the user behavior model is estimated and stored. The pre-training task estimates the parameters of the deep learning model using the historical data. The fine-tuning adjusts the parameters based on the information from the preceding day. The current data (i.e., t_0) is then used in the detection of the threat. To achieve this objective, the architecture performs two sets of analysis: word prediction and anomaly detection.

V. Case Study

Data engineering pipeline. From the data tables available in the Los Alamos cybersecurity events dataset [7] we select *authentications* as the source for the analysis. Figure 3 shows the elements of an authentication record.

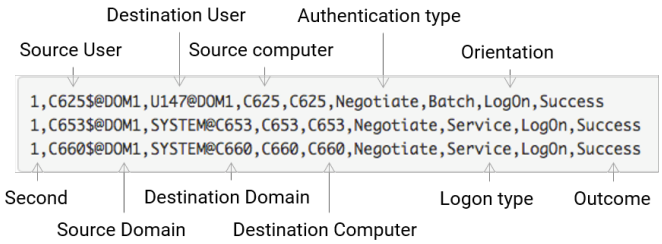


Fig. 3. Transforming authentications to sequences.

There are 1.051 billion authentication records in the dataset. Each record includes the time (in seconds) when it happened, as well as other key data such as the users, computers, type and direction involved in the authentication. As can be observed in Figure 3, there are multiple authentication events in any given time instance which involve different entities. Through the pre-processing stage we transform the dataset and obtain the 'words' to be used in downstream processes.

The next stage is the creation of sequences, first into 'sentences' and later on into 'documents' where sentences are separated by the special token [SEP]. To build the sentence structure we consider the most optimal, meaningful information that will enable the threat detection. Through experimentation we find that the maximum number of tokens that support short pre-training and fine-tuning cycles is 256. Conversely, a 'short' sentence may contain less information but enable the use of longer sequences and still succeed at detecting the insider threat. A review of the data shows that the user behavior on any given second (which in NLP is analogous to a document) may contain anywhere from 250 to 3,000 tokens (i.e., words), with the median around 2,000. We select 2048 as the maximum sequence length. Upon tokenization, we determined that the total number of unique events – the vocabulary size is around 30K. The data is now ready to be used by the DSML pipeline.

DSML pipeline As mentioned before, we adopted BERT as the deep learning architecture for the detection of the threat. The original BERT base design was composed of 12 layers (i.e., stacked transformer encoders) using 768 dimension for the hidden states and 12 attention heads. The total number of trainable parameters is 109M, equivalent to all the weights that will be adjusted during training. Given the time constraints that we have articulated, and the efficiency we require in the process to enable fast response, we reduce the model size by decreasing the number of heads and encoders to six and the hidden state size to 384. Although the vocabulary size is

Len	Sequence	Cumulative probability >50%	[MASK]	Ground truth
565	... hour10 weekday3 [MASK] C585 C585 ...	U2753	U2753	Normal
216	... hour4 weekday3 U8 [MASK] C423 ...	C568 C62	C62	Normal
85	... hour8 weekday4 U77 C616 [MASK] ...	C2742 C616	C616	Normal
693	... hour10 weekday1 U1506 [MASK] C2388 ...	C2388 C3610	C17693	Threat
397	... hour7 weekday1 U8946 [MASK] C19038 ...	C19038	C17693	Threat
389	... hour18 weekday2 U8840 C17693 [MASK] ...	C625 C612 C467 C586 C457	C370	Threat

Fig. 4. User behaviours prediction for insider threat detection.

comparable to that of the English language, the short sentence structure has just five unique types of entities: *computers, users, hours* and *weekdays*. In contrast, a natural language uses myriad different token types. This simpler architecture uses approximately 30M trainable parameters, driving contained and efficient resource usage while still representing each token by rich 384-dimensional vectors. We proceed with the pre-training of the model using 14-day historical data, or 1,209,600 documents (i.e., seconds). When using an NVIDIA V100 GPU and the short sentence structure the pre-training takes approximately 40 hours for 4 epochs. The fine tuning of the user behavior model is performed using the data from the preceding day. As expected, resuming the training with the latest daily data can be done quite rapidly (e.g., overnight). The user behavior model is now suitable for use by the core detection processes.

Insider threat detection. The user behaviour model learned in the pre-training stage can also be used for sequence prediction. A key advantage of a BERT-centric architecture is the ability to perform transfer learning. Any token in the sequence can be masked and predicted with the model. We select the current second (t_0) in Figure 2, pre-process the raw data and create the sequence to be inputted in the model. Figure 4 displays six different instances for analysis.

The first three sequences correspond to normal user behaviors captured in the data. The first sequence has 565 tokens, and it is entered into the model masking the user name. All predicted values whose probabilities add up to more than 50% are displayed. For the first sentence, only one value is needed for surpassing the 50% threshold; this is U2753 with a probability of 90.56%. The model is quite certain about the prediction, and the actual value is indeed U2753. Therefore, we can consider the behavior as normal.

The second sequence has 216 tokens in it, and we now mask the source computer. The two highest-probability predictions cumulatively reach the 50% threshold: C568 and C62. The actual source computer used by the user U8 was C62, so the model is once again correct with no indication of an insider threat taking place. The third sequence is very short, just 85 tokens, and we mask the destination computer. As it was the case in the preceding one, the correct computer C616 is predicted, so we can consider the behavior a normal one.

We now proceed to analyze three known *threat cases*. It is important to emphasize that this ground truth data is used to evaluate the accuracy of the prediction, but was not used when training the model. In the first case we mask the source computer for user U8946 in a 693-tokens sequence. Two computer predictions accumulate beyond 50%: C2388 and C3610, but none of them is the actual computer in the data, i.e., C17693. In this case, none of the predictions ended up being correct. This is an indicator of an insider threat that would be communicated to a human for further review. The incorrect prediction by the model is a correct indicator of an insider threat. The fifth sequence is 397-tokens long, and we mask again the source computer. The model in this case is also strongly convinced that the source computer must be C19038 (with a very definitive 99% probability). However, the actual source computer is different, which is a clear indicator of an abnormal event, as the model has high confidence in a prediction that ends up being incorrect. In the last case we mask the destination computer and enter the data in the model. Many predictions are needed to reach the 50% threshold – which can be interpreted as the model having difficulties to predict with a high-level of certainty. The actual value C370 is not in the prediction group, which again points to a potential insider threat and shall be sent to a human for review.

VI. Conclusion

This research formulates the insider threat detection as an attention-based machine learning problem. The objective is to effectively detect a threat taking place with optimal efficiency driving a timely response from a human actor. We demonstrate how a Transformer deep learning configuration based on the BERT architecture achieves this objective by leveraging the strengths of an attention-based configuration. We demonstrated how to identify a potential insider threat in near real-time using a well-defined three-step machine learning process.

References

[1] AI index report 2021. https://aiindex.stanford.edu/report/, 2021.
[2] S. Barot. Realizing Value From Composable AI Through XOps.
[3] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-Based Models for Speech Recognition. *arXiv:1506.07503 [cs, stat]*, June 2015.
[4] D. L. Costa, M. J. Albrethsen, M. L. Collins, S. J. Perl, G. J. Silowash, and D. L. Spooner. An Insider Threat Indicator Ontology. page 87.
[5] Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019.
[6] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic, Dec. 2013. IEEE.
[7] A. D. Kent. Comprehensive, Multi-Source Cybersecurity Events, 2015.
[8] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[9] Liu et al. Anomaly-Based Insider Threat Detection Using Deep Autoencoders. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 39–48, Nov. 2018.
[10] E. Lopez and K. Sartipi. Feature Engineering in Big Data for Detection of Information Systems Misuse. page 12, 2018.
[11] E. Lopez and K. Sartipi. Detecting the Insider Threat with Long Short Term Memory (LSTM) Neural Networks. *arXiv:2007.11956 [cs]*, July 2020.
[12] S. Paul and S. Mishra. LAC: LSTM AUTOENCODER with Community for Insider Threat Detection. In *2020 the 4th International Conference on Big Data Research (ICBDR'20)*, pages 71–77, Tokyo Japan, Nov. 2020. ACM.
[13] Vaswani et al. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

A Novel Network Alert Classification Model based on Behavior Semantic

Zhanshi Li[a], Tong Li[a,*], Runzi Zhang[b], Di Wu[a], Zhen Yang[a]
[a] Beijing University of Technology
[b] NSFOCUS Technologies Group Co., Ltd., Beijing 100089, China
*litong@bjut.edu.cn

***Abstract*—With the ever-increasing complexity and scale of today's information systems, security personnel have to face massive alerts every day. Efficiently and precisely classifying such alerts in terms of different threat levels is essential for protecting systems from serious threats. However, existing works mainly focus on binary classification which are coarse-grained and cannot pragmatically reduce the number of alerts examined by security personnel. This challenge is further exacerbated when dealing with large numbers of alerts. Moreover, existing approaches only focus on textual and temporal features without considering behavior semantic information when training classifiers. In this paper, we propose a behavior semantic-based alert multi-classification model. Specifically, our proposal classifies alerts into three threat levels according to the real practice of our industrial collaborator. In the meantime, we extract attribute, temporal, and behavioral semantic features to help classifiers to better learn classification boundaries. To extract behavior semantic information, we embed the alert attributes by learning their contextual behavior semantics and use the embedding vector as behavioral semantic features. We conducted experiments using a real enterprise network alert dataset. The experimental results demonstrate that our approach outperforms baseline methods in terms of false positive rate and classification performance.**

***Index Terms*—Alert Classification, Semantic Feature, Machine Learning**

I. Introduction

In large enterprises, security personnel need to continuously monitor, analyze, and classify security alerts to defend against malicious attacks [1]. Security alerts are typically classified into multiple levels by security personnel. However, modern large-scale systems face a vast number of security alerts every day, desperately requiring an effective and efficient multi-classification approach. While there has been some work exploring the task of alert classification, it still has shortcomings.

Firstly, existing works [2], [3] mainly focus on binary classification which are coarse-grained. Coarse-grained alerts cannot pragmatically reduce the number of alerts examined by security personnel. Especially, when dealing with a huge number of alerts, existing approaches cannot effectively reduce the workload of security personnel. For example, Lin et al. [2] classify alerts into alerts triggered by attacks and by non-attacks, respectively. They believe that alerts triggered by attacks are all important high-threats. However, this is not the case in practice. Because some attacks (e.g., worm attacks) are low-threat and do not cause actual loss to the enterprise. The security personnel's real needs are identifying high-threats, which are usually not that many and feasibly to be manually reviewed. In conclusion, the alert classification method based on binary classification is not reasonable.

Secondly, existing approaches only focus on textual and temporal features without considering behavior semantic information when training classifiers. Specifically, these methods are simple in representing alert attributes without considering the behavior semantic information of the alert attributes. For example, Shittu et al. [4] represent IP addresses as binary strings to calculate the distance between alerts. The distance between IP addresses is measured by the length of the longest common prefix. The way the distance between IP addresses is calculated shows that the longer the common prefix, the closer the logical distance between different IP addresses. However, the longest common prefix length for two IP addresses only implies whether they are in closer subnets and does not indicate similar behavior.

To address the above shortcomings, we propose A behavioR seMantic-based alert multi-Classification mOdel (ARMCO). Specifically, ARMCO extracts a series of alert features from the attributes of historical alerts, including attribute, temporal, semantic features. Attribute features encode discrete attributes of alerts with a wide range of values into a low-dimensional vector space, which save space and facilitates subsequent classification model learning. Temporal features can characterize the temporal aspects of alerts for different threat levels. To extract behavioral semantic information, we embed the alert IP addresses and ports by learning their contextual behavior semantics, and use the embedding vector as semantic features. Therefore semantic features can characterize the contexts of IP addresses and ports at different threat levels alerts. In the meantime, the representation of the IP addresses and ports is kept consistent with their behavior characteristic. In addition, ARMCO classifies alerts into high-threat, low-threat, and non-threat alerts according to the real practice of our industrial collaborator. Fine-grained classification of alerts more conform to the real network situation, and more friendly to security personnel as well. The major contributions of this paper are the following:

- We classify alerts into non-threat, low-threat, and high-threat, which are more in accordance with the threat situation of real network alerts.
- We embed the alert attributes by learning contextual

DOI reference number: 10.18293/SEKE2022-116.

behavior semantics to consistent their representation and behavioral characteristics.
- We conducted a number of experiments on a real network dataset, the results of which demonstrate the effectiveness of ARMCO.

II. Related Work

Recently there is a lot of work dedicated to the alert post-processing, such as alert correlation [5]–[7], aggregation [8], clustering [9]. These works are interdependent, and the dividing line is blurred. However, they aim to reduce the number of alerts or detect the attack. Shittu et al. [4] extracted the association conditions from historical alerts by using posterior probabilities. They use these conditions to correlate alerts, called meta-alert. Finally, the different local outliers of meta-alert are mapped to different alert threat levels. Low-threat alerts can then be filtered. Hofmann et al. [10] proposed an online intrusion alert aggregation system, which uses a finite mixture distribution to measure the similarity between alerts. However, the effectiveness of this approach depends on the assumptions of the distributions. Siraj et al. [11] design domain-specific similarity calculation methods for different alert attributes. The similarity of two alerts is a weighted sum of the similarity of attributes. Then, the alerts are aggregated by similarity for attack detection. Ahmadinejad et al. [12] use some temporal windows to reduce the comparison of new alerts with the full history alerts. After that, they use a probability-based evaluation function to determine the threshold value of the alert association. Two alerts are associated if their temporal similarity is higher than the threshold value. The correlated alerts are treated as hyper-alerts, which can be used for attack detection. In summary, the above works aim to relieve the pressure on security personnel. However, many alerts still need to be examined, and security personnel cannot focus on examining high-threat alerts.

Identifying the threat level of alerts is a relevant research topic in intrusion detection systems (IDS) [13]. Shittu et al. [4] mines possible correlation conditions in the alerts history and set correlation thresholds for different correlation conditions. Two alerts can be correlated when they satisfy multiple correlation conditions simultaneously. Directly or indirectly related alerts form a correlation graph. High-threat correlation graphs are discovered by calculating the similarity among correlation graphs. Lin et al. [2] jointly model alert temporal dependencies and textual dependencies to discover actual attacks in alerts. Temporal dependence exploits Bayes' rule and prefix tree to extract attack patterns. Text dependencies measure the co-occurrence probability between attributes. They combine the two dependencies to sort the alerts in order of probability of being an actual attack. Chen et al. [14] encode and decode high-threat alerts, and the model computes a higher reconstruction error when non-high-threat alerts appear. More specifically, they identify high-threat alerts by two encoding and decoding layers. The above work can also not relieve the pressure of security personnel very well. They didn't make a finer division of non-high-threat alerts, which is unreasonable. In addition, they capture inadequate factors that influence the threat level of alerts.

III. A Behavior Semantic-based Alert Multi-classification Model

Fig. 1 illustrates the overall framework of the ARMCO. As shown in Fig. 1, alerts are input to the semantic, attribute and temporal feature extraction modules. After the three feature extraction modules are computed, their outputs are concatenated to generate the feature vectors. Then the alerts feature vector, including multi-dimension features, is used to train the classification model. Finally, the trained model is used for alert threat level classification. After threat level classification, alerts will be classified as high-threat, low-threat, and non-threat.

A. Extract Attribute Features

Most alert attributes are discrete variables such as source IP address (Sip), destination IP address (Dip), source port (Sport), destination port (Dport), the rule id utilized to trigger the alert (RuleId) and the attack category to which the traffic belongs (AttackType). The above alert attributes include important information about an attack, which is useful for classifying the alert threat level. To facilitate the learning of the classification model, these discrete variables need to be encoded as feature vectors. In this paper, we use hash coding to encode discrete attributes. The hash encoding approach is more suitable for cases where the variables have a large range of values. And it can map variables with any range of values to feature vectors of a small length. Let the set of values of a variable v be V. v including n different values, we use hash coding approach to map a ($a \in V$) to a feature vector $\vec{a}$ of length $\sqrt{n}$. Specifically, we use the signed 32-bit variant of MurmurHash3 as hash function. MurmurHash3 is an extensively tested and fast non-cryptographic hash function [15]. It has good distributivity and is suitable for machine learning. The RuleId and AttackType attributes of the alert can be encoded directly as described above. The remaining four attributes need to be coded in two separate groups, Sip, Dip and Sport, Dport, respectively. Because Sip and Dip belong to the IP address, they need to be encoded together, so do Sport and Dport. The following description is how to encode Sip and Dip, the same for Sport and Dport. Let the set of values of Sip be S, the set of values of Dip be D,and $V = A \cup D$, and $n = |A| + |D|$. Then for b ($b \in V$) and n, they also input above-mentioned hash function to get the feature vector.

B. Extract Temporal Features

In this section, the construction of temporal features is described. For simplicity, no-threat, low-threat and high-threat are represented by 0, 1 and 2 respectively.

Alert count. We define the number of alerts in a specific time window as the alert count (e.g., the total number of alerts in the 5 minutes before the current alert was triggered). Intuitively, security personnel should pay more attention to an alert with a sharp change in the number of alerts in a short period [16].

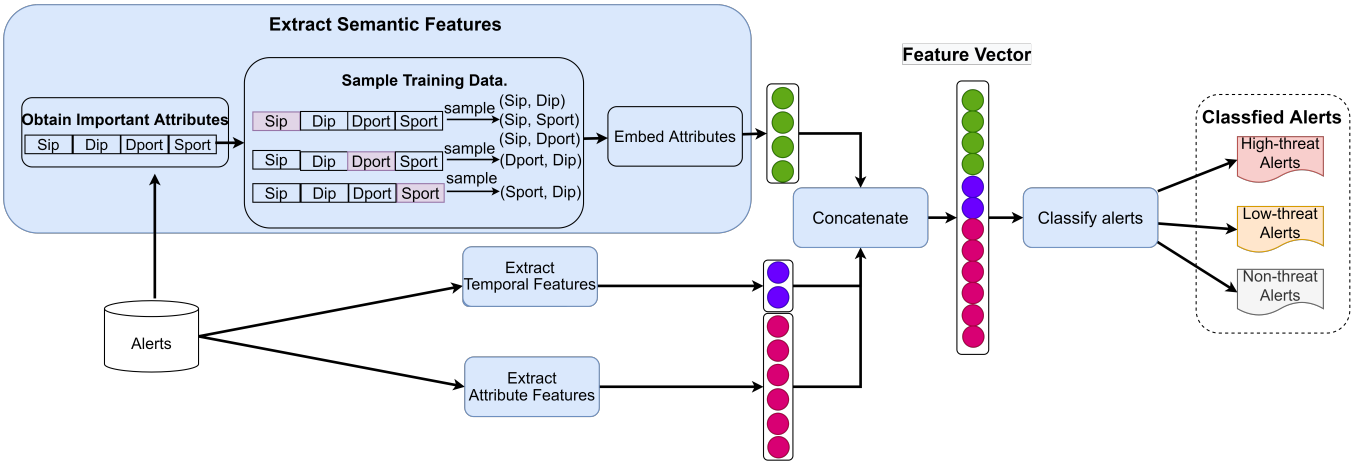


Fig. 1. The architecture of ARMCO.

Inter-arrival time. We define the time difference between the current alert and the previous alert as the inter-arrival time [16] (e.g., if alert A was triggered at 2021.03.22 00:00:00 and alert B was triggered at 2021.03.22 00:01:00, the inter-arrival time for alert B is 60 seconds). Intuitively, the first alert after a long period of no alerts should be focused on by security personnel [16].

C. Extract Semantic Features

For simplicity, we consider Sip, Dip, Sport and Dport as important attributes of the alert. Although the important attributes are also encoded by the attribute features, they are encoded for different purposes. The attribute features are designed to numerically encode discrete attributes.And semantic features aim to encode important attributes with contextual behavioral semantics. The contextual behavioral semantics of attributes refers to different IP addresses or ports that are similar if they appear in similar contexts. The approach to embedding important attributes of alerts follows the idea of IP2Vec [17] but is still somewhat different. While IP2Vec embeds IP addresses in network traffic, our approach embeds both IP addresses and ports in alerts. We extract the context of IP addresses and ports from historical alerts and construct training samples to input the embed attributes module. Fig. 1 shows the architecture of embedding the IP addresses and ports. Several key tasks are explained in detail next.

Selection of context. How to select behavioral semantics of important attributes is a core issue. As introduced in Section I, existing methods are simple in their representation of alert attributes and fail to capture the differences between their behaviors. Each alert describes the information related to specific anomalous traffic, as shown in Table I. Not all attributes are helpful. It is not difficult to find that the important attributes themselves are closely related because host communication requires this information. Therefore, we choose the important attributes themselves as their context.

Sampling of training data. The Sip, Dip, Sport and Dport of each alert are considered a ”sentence”, we use a subset of the ”sentence” to construct the input and output words. As shown in Fig 1, it shows how to sample training data. When using Sip as input words, Dip, Sport, and Dport are used as context words. When Dport is used as the input word, only Dip is selected as the context word. When using Sport as the input word, only Dip is selected as the context word. After the above sampling, input-output word pairs can be obtained, which are inputted into the module of embedding attributes.

Embed attributes. After receiving the input-output pairs, we need to convert them into knowledge graphs. The nodes in the knowledge graph are the set of all inputs and outputs. For each input-output pair, a directed edge from input to output is added to the knowledge graph. There are at most two edges with different directions between two nodes. In the module of embedding attributes, we adopt interactE([18]) to embed the attributes. InteractE is based on three key ideas – feature permutation, a novel feature reshaping, and circular convolution. The interactE optimizes the performance of embedding by increasing feature interaction.

D. Classify alerts

After feature extraction, each alert is represented by a set of features, and each alert also includes a threat level label that is examined and labeled manually. The XGBoost classification model is adopted for the alert threat level classification task. XGBoost is a gradient boosting tree-based model that is widely used by data analysts and has achieved good performance on many problems [19]. The algorithm creates and combines a large number of separate weak but complementary classifiers to produce a powerful estimator. This combination can be done in two ways: bagging (random forests) and boosting. Gradient boosting is established sequentially. In fact, a new weak learner is constructed to have a maximum correlation with the negative gradient of the loss function of the entire set in each iteration [20]. XGBoost belongs to a group of widely used tree learning algorithms [21]. Decision trees allow prediction of output variables based on a series of rules arranged in a tree structure. They consist of a series of segmentation points, or nodes, determined according to the values of the input features. The last node is a leaf that gives us the specific value of

TABLE I
AN EXAMPLE ALERT.

Attribute	Value
Timestamp	2021-03-22 00:00:00
AlertLevel	2
RuleId	18622209
AttackType	Directory traversal
Sip	*.*.*.*
Dip	*.*.*.*
Sport	48045
Dport	80
Payload	\x085q\x1024\x9cW\xadopu\x08...
q_body	POST /login/login.htm...
r_body	HTTP/1.1 200 OK \r\n Server...

the output variable. Tree learning algorithms do not require linear features or linear interactions between features. They are significantly better classifiers than other algorithms [22]. In addition, XGBoost, a gradient boosting algorithm, has two major improvements: accelerated tree construction and proposed a new distributed tree search algorithm.

IV. EXPERIMENT

In this section, we use real network dataset to evaluate ARMCO, aiming to answer the following research questions:

- RQ1: Can ARMCO effectively improve the alert threat level classification performance?
- RQ2: How much do different features affect classification performance?

A. Datasets

We first present information about the attributes of the alerts and the dataset used.

1) Alert Description: an alert, which has multi-dimensional alert attributes, is the smallest core data structure we study and analyze. Table I presents an example alert with several major attributes. The Timestamp attribute indicates the time when the alert was triggered. The AlertLevel attribute indicates the threat level determined by the traditional rule-based approach. The higher the value, the higher the threat level. The RuleId attribute indicates the rule ID utilized to trigger the alert. The AttackType attribute indicates the attack category to which the traffic belongs. The Sip, Dip, Port, and Dport attributes indicate the source IP address, destination IP address, source port, and destination port of the traffic, respectively. The payload attribute indicates the alert payload, including IP layer and lower layer data. The q_body attribute indicates the request body of the web access. The r_body attribute indicates the response body of the web request.

2) Dataset Description: the network alert dataset used in this paper is collected from a real security company. This dataset records a day's total of about nine million alerts. As Table I shows, while each alert log includes its threat level, that threat level is evaluated by a rule-based approach that has poor classification performance. For further analysis and to get better classification performance, the security personnel examine each alert and label it with the real corresponding threat level.

B. Metrics

As described in Section III, the trained classification model classify the test set alert. After all the alerts are classified, the precision (P) /recall (R) /F1-score (F1) /false positive rate (FPR) be calculated to evaluate the performance of each class. Each class can calculate P, R, F1 and FPR. Finally, we use the weighted P, R, F1 and FPR to evaluate the model performance. The weight of each class is equal to its proportion in the test set. Precision measures the percentage of identified threat levels that are the same as the actual threat level. Recall measures the percentage of alert threat levels that are correctly identified. F1-score is the harmonic mean of precision and recall. FPR measures the percentage of alert threat levels that are incorrectly identified. Therefore a better classification model has a higher P, R, F1 and a lower FPR.

C. Parameter Settings

We tune the hyperparameters of the ARMCO in the validation set (split from trainset). In the attribute features, the length of the feature vectors after hashing Sip, Sport, Dip, Dport, RuleId and AttackType are 24,124,24,124,5,5, respectively. In the temporal features, the window size of the alert count is set to 165 seconds. In interactE model, the attributes are embedded with a feature vector dimension of 32. For each positive sample, 50 negative samples are randomly sampled. Also, we set input dropout rate to 0.2, feature dropout rate to 0.5, hidden dropout rate to 0.5. It is trained via stochastic gradient descent over shuffled mini-batches with a batch size of 128. It uses an Adam optimizer with a learning rate is 0.0001. In the XGBoost classification model, we set lambda to 2, gamma to 0.1, max_depth to 6, subsample to 0.6, colsample_bytree to 0.9, min_child_weight to 3 and eta is 0.1.

D. Experimental Settings

To answer the research questions raised above, we designed two experiments. The following experiments uses the dataset introduced in Sec. IV-A. We use the top 80% of the alerts that have been sorted in time order as the training set and the remaining 20% as the test set.

1) Experiment 1: To demonstrate the performance of the ARMCO, we compare it with two baseline methods. The compared methods are listed below.

- Rule-based. The traditional rule-based approach classifies alerts into different threat levels (e.g., high-threat, low-threat, and non-threat). Experienced security personnel is required to keep the rule database updated regularly.
- Bug-KNN [23]. Bug-KNN calculates the similarity between bugs by textual similarity. It uses K-Nearest Neighbor to calculate the distribution of severity levels in historical bug reports most similar to the new bug report.

2) Experiment 2: The features ARMCO extracted include the attribute, temporal, semantic features. To demonstrate the effect of different features on the performance, we prepare three variants of ARMCO: $ARMCO_{TS}$, $ARMCO_{AT}$ and $ARMCO_{AS}$. They are differentiated below.

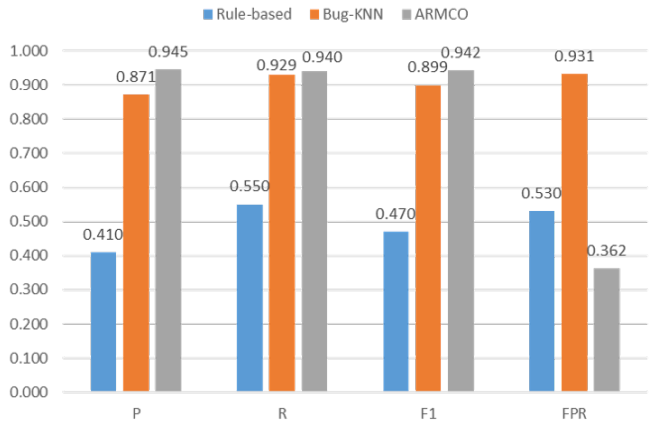


Fig. 2. Performance comparison of different methods.

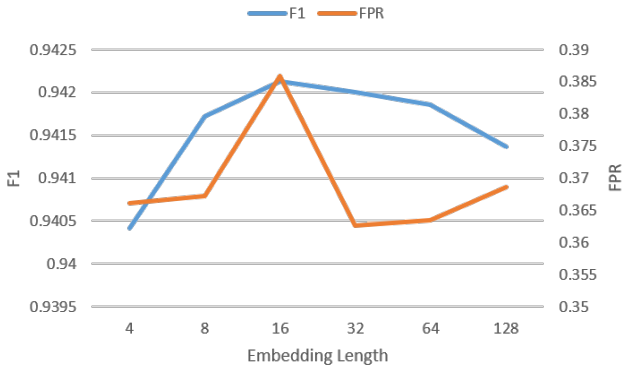


Fig. 3. Effects of embedding size.

TABLE II
EFFECT OF DIFFERENT FEATURE.

	P	R	F1	FPR
$ARMCO_{TS}$	0.944	0.939	0.941	0.366
$ARMCO_{AT}$	0.937	0.930	0.933	0.406
$ARMCO_{AS}$	0.892	0.666	0.750	0.497
ARMCO	**0.945**	**0.940**	**0.942**	**0.362**

- $ARMCO_{TS}$: This variant removes the attribute features.
- $ARMCO_{AT}$: This variant removes the semantic features.
- $ARMCO_{AS}$: This variant removes the temporal features.
- $ARMCO$: This is our complete model which involves all the proposed features.

E. Experimental Results

1) Experiment 1: Fig.2 shows the performance comparison of ARMCO and baseline methods in P, R, F1 and FPR. ARMCO outperformed other baselines, achieving an F1 of 0.942, higher than others, and an FPR of 0.362, lower than others. Compared with Rule-based, ARMCO's F1 is increased by 100%, and FPR is reduced by 31.6%. Compared with Bug-KNN, ARMCO's F1 increased by 4.7% and FPR reduced by 61.1%. As shown in Fig.2, the Rule-based approach has an F1 of 0.470, FPR of 0.530, which is weakly effective. Due to various factors influencing the threat level of alerts, using rules alone can't completely capture these factors. Besides, the Rule-based approach requires security personnel to spend plenty of time and effort to maintain the rules.

Bug-KNN measures the threat level by calculating the threat distribution of the historical alerts that are most similar to the current alert. As shown in Fig.2, Bug-KNN has an F1 of 0.899, FPR of 0.931. The results show that the Bug-KNN is weakly effective and has a very high time complexity. Due to various factors influencing the threat level of alerts, using textual information alone can't completely capture these factors. In summary, the experimental results demonstrate the effectiveness of ARMCO on classifying the alert threat level.

2) Experiment 2: Table II shows the performance of ARMCO with its three variants. From Table II, we can see that ARMCO can achieve the highest F1 of 0.942 and the lowest FPR of 0.362 when all features are used. The following is a detailed introduction to the impact of each ARMCO variant on performance.

- $ARMCO_{TS}$: When this variant was used, F1 drops from 0.942 to 0.941, and FPR increases from 0.362 to 0.366 compared to ARMCO. The phenomenon shows that attribute features have little effect on improving F1 and reducing FPR. Therefore, attribute features have weakly impact on improving classification performance.
- $ARMCO_{AT}$: When this variant was used, F1 drops from 0.942 to 0.93, and FPR increases from 0.362 to 0.406 compared to ARMCO. Compared with the attribute features, the semantic features have a greater effect on improving F1 and reducing FPR. Semantic features can be seen as a way to encode alerting attributes. Semantic features encode only four attributes. Due to the encoding process considering the semantic information of the attributes, the performance leads to significant improvement. This experimental result demonstrates the effectiveness of the semantic feature and reflects our second contribution.
- $ARMCO_{AS}$: When this variant was used, F1 drops from 0.942 to 0.75, and FPR increases from 0.362 to 0.49 compared to ARMCO. Compared with the previous three features, the temporal features have the biggest effect on improving F1 and reducing FPR. This indicates that the temporal features significantly affect classification performance.

In summary, the experimental results show that different features have different extent effects on the performance. And the temporal feature has the greatest effect, followed by semantic feature, and attribute feature.

F. Effects of Parameters

Our model has important parameters that need to be tuned. Here, we evaluate the impact of one parameter on performance, i.e., the embedding size in semantic features.

To investigate the effect of the embedding size in semantic features, we vary it in the set of 4, 8, 16, 32, 64, 128 and record performance results. From Fig. 3, we can see that the

F1 first improves with the increase in embedding size, and then starts to decrease at size 16. From Fig. 3, we can see that the FPR first improves with the increase in embedding size and then starts to decrease at size 16, finally starts to increase at size 32. As mentioned in Section IV-B, we expect larger F1 and smaller FPR. When the embedding size is smaller, the alert's important attributes may overlap in the low-dimensional representation space and cannot be represented accurately, so F1 and FPR are a bit worse. When the embedding size is larger since the behavioral semantics of the alert's important attributes are stationary, it leads to a deviation of the learning direction during the embedding process, so F1 and FPR are a bit worse. Although F1 achieves a maximum of 0.9422 at size 16, the corresponding FPR is high at 0.3859. FPR achieves a minimum of 0.362 at size 32, and although the corresponding F1 of 0.942 is 0.0002 lower than the maximum, it is ignorable. So the optimal performance can be obtained at the embedding size 32.

V. Conclusion and Future Work

In this paper, we propose a behavior semantic-based alert multi-classification model, which contains a set of powerful features to measure alerts of different threat levels. The results demonstrate the effectiveness of ARMCO and achieve an F1 of 0.942 and an FPR of 0.362. This dramatically reduces the pressure on security personnel to examine and minimize losses to the enterprise.

Alerts do not occur in isolation. Some attacks require relatively fixed attack steps to exploit the target, and different attack steps trigger different alerts. This can lead to the alert under different alert contexts having different threat levels, and how to incorporate this information into the alert threat assessment needs further study. Besides, we will also evaluate the proposed approach to real-time scenarios in future work.

Acknowledgement

This work is partially supported by the Major Research Plan of National Natural Science Foundation of China (92167102), the Beijing Nova Program (Z211100002121150), the Project of Beijing Municipal Education Commission (No.KM202110005025), and Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education.

References

[1] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2016, pp. 102–111.

[2] Y. Lin, Z. Chen, C. Cao, L.-A. Tang, K. Zhang, W. Cheng, and Z. Li, "Collaborative alert ranking for anomaly detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1987–1995. [Online]. Available: https://doi.org/10.1145/3269206.3272013

[3] M. E. Aminanto, T. Ban, R. Isawa, T. Takahashi, and D. Inoue, "Threat alert prioritization using isolation forest and stacked auto encoder with day-forward-chaining analysis," *IEEE Access*, vol. 8, pp. 217 977–217 986, 2020.

[4] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and M. Rajarajan, "Intrusion alert prioritisation and attack detection using post-correlation analysis," *Computers & Security*, vol. 50, pp. 1–15, 2015.

[5] S. A. Mirheidari, S. Arshad, and R. Jalili, "Alert correlation algorithms: A survey and taxonomy," in *International Symposium on Cyberspace Safety and Security*. Springer, 2013, pp. 183–197.

[6] A. A. Ramaki, M. Khosravi-Farmad, and A. G. Bafghi, "Real time alert correlation and prediction using bayesian networks," in *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. IEEE, 2015, pp. 98–103.

[7] S. Salah, G. Maciá-Fernández, and J. E. Díaz-Verdejo, "A model-based survey of alert correlation techniques," *Computer Networks*, vol. 57, no. 5, pp. 1289–1317, 2013.

[8] D. Man, W. Yang, W. Wang, and S. Xuan, "An alert aggregation algorithm based on iterative self-organization," *Procedia Engineering*, vol. 29, pp. 3033–3038, 2012.

[9] D. Lin, R. Raghu, V. Ramamurthy, J. Yu, R. Radhakrishnan, and J. Fernandez, "Unveiling clusters of events for alert and incident management in large-scale enterprise it," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1630–1639.

[10] A. Hofmann and B. Sick, "Online intrusion alert aggregation with generative data stream modeling," *IEEE transactions on dependable and secure computing*, vol. 8, no. 2, pp. 282–294, 2009.

[11] A. Siraj and R. B. Vaughn, "Multi-level alert clustering for intrusion detection sensor data," in *NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE, 2005, pp. 748–753.

[12] S. H. Ahmadinejad and S. Jalili, "Alert correlation using correlation probability estimation and time windows," in *2009 International Conference on Computer Technology and Development*, vol. 2. IEEE, 2009, pp. 170–175.

[13] K. Alsubhi, E. Al-Shaer, and R. Boutaba, "Alert prioritization in intrusion detection systems," in *NOMS 2008-2008 IEEE Network Operations and Management Symposium*. IEEE, 2008, pp. 33–40.

[14] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iboat: Isolation-based online anomalous trajectory detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 806–818, 2013.

[15] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1113–1120.

[16] N. Zhao, P. Jin, L. Wang, X. Yang, R. Liu, W. Zhang, K. Sui, and D. Pei, "Automatically and adaptively identifying severe alerts for online service systems," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2420–2429.

[17] M. Ring, A. Dallmann, D. Landes, and A. Hotho, "Ip2vec: Learning similarities between ip addresses," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 657–666.

[18] S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, and P. Talukdar, "Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 03, 2020, pp. 3009–3016.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[20] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[21] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.

[22] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[23] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," *Journal of Systems and Software*, vol. 117, pp. 166–184, 2016.

Analyzing Cyber-Physical Systems with Learning Enabled Components using Hybrid Predicate Transition Nets

Xudong He
Knight Foundation School of Computing and Information Sciences
Florida International University
Miami, USA
hex@cs.fiu.edu

***Abstract* — Cyber-physical systems (CPSs) are ubiquitous and are becoming increasingly important in the functioning of our society. CPSs have complex discrete and continuous behaviors. In recent years, learning enabled components (LECs) built using machine learning approaches are increasingly used in CPSs to perform autonomous tasks to deal with uncertain and unfamiliar environments. CPSs with LECs are even more difficult to develop. We have developed a methodology for formally modeling and analyzing CPSs with LECs. Hybrid predicate transition nets (HPrTNs) are used as the underlying formal method to model CPSs with LECs and their training through their simulation capability. In this paper, we present our new analysis methodology for CPSs with LECs consisting of three complementary techniques, including a testing technique based on HPrTN simulation capability, a simulation guided barrier certificate technique, and a SMT based bounded model checking technique. The above analysis methodology is partially supported by a tool chain and is demonstrated through an example.**

Keywords — cyber-physical systems; learning enabled components; formal methods; hybrid predicate transition nets; barrier certificate; bounded model checking

I. INTRODUCTION

Cyber-physical systems (CPSs) are ubiquitous and are becoming increasingly important in the functioning of our society. CPSs are hybrid systems that contain physical devices having continuous dynamics and computational control processes with discrete behaviors. These systems are extremely difficult to build and error-prone. In recent years, CPSs have started to use learning enabled components (LECs) as part of the control loop for performing various perception-based autonomy tasks. These data-driven components are trained using machine learning (ML) approaches such as deep learning – deep neural nets (DNNs) and reinforcement learning (RL) [17]. These approaches have provided CPSs the capability to continuously learn and work in uncertain and unfamiliar environments. Although many ML techniques have been developed in the past few decades and tremendous progresses have been made in the last decade, there is very little understanding of the properties of these data-driven models built using ML. Research on the formal analysis of these data-driven models has just emerged in recent years. LECs have added additional dimensions of difficulties to those of CPSs.

We have developed a methodology for modeling and analyzing CPSs with LECs, which contains the following new results: (1) A method for modeling deep neural nets (DNNs) using hybrid predicate transition nets (HPrTNs), (2) An reinforcement learning (RL) technique to train DNNs with an environment (plant) using HPrTNs, (3) A Simplex architecture to integrate advanced controller (a trained DNN) with a baseline controller defined using ordinary differential equations such that the overall system has a closed loop dynamics, (4) A simulation analysis method based on the dynamic semantics of HPrTNs and supported in tool PIPE+, (5) A barrier certificate analysis technique based on inductive invariant reasoning supported in tool Pyomo with linear program solver Gurobi and SMT solver Z3, (6) A bounded model checking analysis approach supported by tool dReach and backend solver dReal. We have presented our detailed modeling method that covers results (1) to (3) in [7]. In this paper, we will provide a brief overview of the modeling method while focus on the analysis techniques covering results (5) to (6). In the following sections, we provide some background information on the modeling method and the details of the analysis techniques.

II. HYBRID PREDICATE TRANSITION NETS

In this section, a formal definition of HPrTNs [6] is provided.

An HPrTN is a tuple $N = (P, T, F, \alpha, \beta, \gamma, \mu, \lambda, M_0)$, where

(1) $P = P_d \cup P_c$ is a non-empty finite set of discrete places P_d and continuous places P_c (graphically represented by circles and double circles respectively);
(2) T is a non-empty finite set of discrete transitions (graphically represented by bars or boxes), which disjoins P, i.e. $P \cap T = \emptyset$;
(3) $F \subseteq P \times T \cup T \times P$ is a flow relation (the arcs of N);
(4) $\alpha: P \to Type$ associates each place $p \in P$ with a type in $Type$. $Type$ defines the structure of the data the places can hold. The basic types include *String*, *Integer*, and *Real*; and the composite types are defined using Cartesian product and power set;
(5) $\beta: T \to Constraint$ associates each transition $t \in T$ with a constraint. Each constraint is a disjunction $\bigvee_i d_i$ for $i \geq 1$, where each disjunct d_i has a canonical form $pre_i \wedge post_i$ that defines the precondition (enabling condition) and post-condition (output result) of a case of t respectively. The precondition contains only variables appearing in the labels of incoming arcs and the post-condition contains variables appearing in the labels of outgoing arcs;
(6) $\gamma: F \to Label$ associates each arc $f \in F$ with a label in the form of a simple variable x or a set element $\{x\}$;

DOI reference number: 10.18293/SEKE2022-010

(7) $\mu: P_c \to (\mathcal{R} \times \mathcal{R})^n$ associates each continuous component of a continuous place with a pair of lower and upper bounds, where n is the number of continuous components;
(8) $\lambda: P_c \to ODE^n$ associates each continuous component of a continuous place an ordinary differential equation that defines its evolution;
(9) $M_0: P \to Token$ is an initial marking and associates each place $p \in P$ with some valid tokens (respecting the type of p and the bounds for continuous components). Each continuous place can only hold at most one token.

The dynamic semantics of HPrTNs are defined based on the markings (states) $M: P \to Token$. A transition $t \in T$ is *enabled* in marking M if one of its precondition is true, Formally: $\forall p \in P. (\theta(\bar{\gamma}(p,t)) \subseteq M(p) \wedge \exists i. (\theta(\beta(t).pre_i)))$, where θ is a substitution that instantiates all the variables in relevant arcs and constraint expression.

An enabled transition $t \in T$ in marking M with substitution θ can *fire*. The firing of transition t results in a new marking M' defined by $\forall p \in P. (M'(p) = M(p) \cup \theta(\bar{\gamma}(t,p)) - \theta(\bar{\gamma}(p,t)))$, which is denoted as: $M \xrightarrow{t/\theta} M'$. The firing of a transition is instant and does not consume time. Two enabled transitions are in conflict if the firing of one of them disables the other. Non-conflict enabled transitions can fire concurrently.

Tokens in continuous places are continuously evolving according to the differential equations governing the change rates as long as their bounds are not violated. Given a marking M, we use $[M]$ to denote the state space covering all possible continuous token evolution with the same token distribution.

Let T_i be a set of concurrently enabled non-conflict transitions with corresponding substitutions θ_i in marking $[M_i]$, and $[M_{i+1}]$ be the resulting new marking after firing T_i with θ_i. The behavior of the net N consists of the set of all firing sequences $[M_0] \xrightarrow{T_0/\theta_0} [M_1] \cdots [M_i] \xrightarrow{T_i/\theta_i} [M_{i+1}] \cdots$. The set of all reachable markings is denoted as $[[M_0] >$.

III. MODELING CPS WITH LECS USING HPRTNS

Our modeling methodology based on HPrTNs consists of three steps: (1) modeling LECs using DNNs, (2) training LECs though modeling environment and system dynamics using reinforcement learning, and (3) modeling and integrating LECs with other conventional system components within the HPrTN paradigm. We briefly discuss our modeling methods in steps (1) and (2) below.

DNNs have become a dominant deep learning approach in recent years. A DNN has an architecture, which consists of an input layer, multiple hidden layers and an output layer. Each layer contains multiple neurons (each is represented by a circle) that contain numerical values. The value of a neuron in layer l is calculated through an activation function on the weighed input from neurons in layer $l-1$. Different types of DNN architecture can be obtained based on how the adjacent layers are connected, including feedforward (fully connected), convolution, and recurrent. DNNs are trained using the output results. A cost function defined on the output is used to calculate the final error rate. By calculating and propagating the error rates layer by layer backwards starting from the final error rate, we can adjust the weights and biases based on the error rates.

We have developed a novel HPrTN template to model a DNN with backpropagation, where the architecture of the DNN is modeled as follows:

(1) Each layer l in DNN is modeled by a discrete place p_l of type $\mathcal{R} \times ... \times \mathcal{R}$, where the cardinality determines the number of neurons within the layer;

(2) Modeling neurons – each neuron is modeled by a token (or a token component) of a real type, and the neurons within the same layer is modeled by a structured token;

(3) Let l and $l+1$ be two layers with cardinality m and n respectively, a discrete place w_l of type $(\mathcal{R}^m)^n$ is used to model the weight matrix between these two layers and a discrete place b_l of type $\mathcal{R}^n$ is used to represent the bias vector;

(4) A transition t_l with input places p_l, w_l, b_l, and output place p_{l+1} is used to model activation function between these two layers, the transition constraint $\bigwedge_{i=1}^{n} z_i = \sigma(w^i x^T + b_i)$ defines the algebraic relationships between the activations (the neurons) in these two layers, where each z_i is a weighted input to neuron i in layer $l+1$;

(5) A transition $cost$ is added with the constraint defining initial error estimation. This transition has the place modeling the final output layer as an input, and an output place for error propagation;

(6) A place $doutput$ is added as an input to the $cost$ transition, a token specifying the desirable output y resides in this place;

(7) A place e_l abstracting backward error propagation is added between layers l and $l+1$;

(8) A transition g_l is added between layers l and $l+1$; this transition produces the backward error and updates the weights and biases of layer $l+1$.

Fig. 1 shows an HPrTN of two adjacent layers of a DNN with backpropagation:

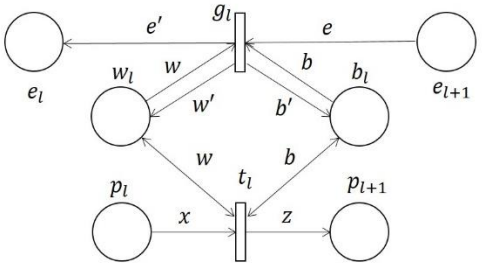


Fig.1. An HPrTN of two layers of a DNN with backpropagation

Reinforcement learning (RL) is a major machine learning approach, which learns how to attain a complex objective (goal) or how to maximize along a particular dimension over many steps. An agent (controller) continuously interacts with an environment (plant). The agent selects some action a according to a policy π defined using a value function on an input state s and reward r or defined using a Q-value function on an input pair of state s and action a. The environment generates a new

state s' and reward r' according to the given action a. The goal is to maximize cumulative rewards when a final state is reached.

Our method is based on neural fitted Q-learning process [10], which builds an HPrTN model for a CPS with LECs and uses the simulation capability of HPrTNs to train LECs modeled as a DNN where a baseline controller or plant is used as the learning environment. We have developed several HPrTN templates to capture temporal difference methods in RL, which support a variety of RL learning settings, including on / off line, model based / model free, stationary / non-stationary, and discrete / continuous.

To demonstrate the applicability of the method to model and train a LEC, we have used the following car system adapted from [1]: a car needs to move along a circular track as closely as possible. The sensors (simulated) of the car can detect the center of the track. The car's position is defined by its coordinates (x, y). The car has a direction θ and a speed v. The car has three modes straight, left, and right and the dynamics in each mode is as follows:

- Right: $\dot{x} = (v cos\theta)/2$, $\dot{y} = (v sin\theta)/2$, $\dot{\theta} = -\pi$, $d \geq e$;
- Straight: $\dot{x} = v cos\theta$, $\dot{y} = v sin\theta$, $\dot{\theta} = 0$, $-e \leq d \leq e$;
- Left: $\dot{x} = (v cos\theta)/2$, $\dot{y} = (v sin\theta)/2$, $\dot{\theta} = \pi$, $d \leq -e$.

A parameter e is used to define the error margin [$-e$, e], and the distance d between the car's current position and the center of the track is calculated dynamically to control the switching between modes. A baseline controller mimicking the environment of the car and several advanced controllers (LECs) using different DNN architectures and activation functions have been tried. Fig. 2 shows a trained advanced controller (AC) together with a baseline controller (BC) modeled using an HPrTN developed in PIPE+ [14].

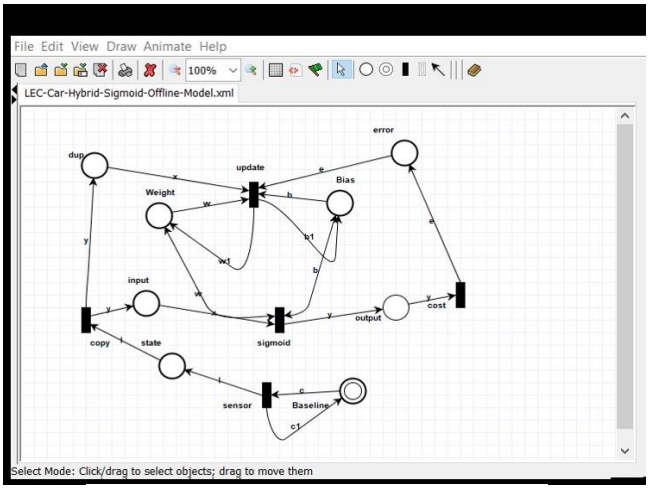


Fig.2. An HPrTN representing a CPS with a LEC

Training is done using HPrTN's simulation capability. For example, we have run six batches with randomly generated with position (x, y), and direction θ. Each batch contains 1000 execution steps. The overall training involves firing 100,000 transitions and takes 31673 milliseconds on a PC with Intel(R) Core(TM) i7-4770S CPU @ 3.10 GHz and 8 GB RAM running Windows 10 OS.

IV. Analyzing CPSs with LECs

Three techniques for analyzing CPSs with LECs are explored, including simulation, barrier certificate, and SMT based bounded model checking. Simulation is supported by the operational semantics of HPrTNs, which can be used to train LECs as well as test CPSs with LECs by selecting targeted or random initial markings. Simulation results also provide the basis for barrier certificate analysis. Simulation is easy to use, scalable, and fully automatic. Simulation is supported by our tool environment PIPE+. In the following sections, we describe the barrier certificate and the bounded model checking techniques.

A. Barrier Certificate Technique

Barrier certificate technique is based on symbolic simulations for finding inductive invariants to prove the safety requirements of a dynamic system. Since a CPS with LECs modeled in an HPrTN is executable and produces simulation traces, we can apply barrier certificate technique to analyze the dynamics of the whole closed loop system.

A barrier certificate is a differentiable function B from the set of states of the dynamical system to the set of real numbers satisfying the following conditions:

(1) $\forall x \in X_0$: $B(x) \leq 0$, where X_0 is the set of possible initial states,

(2) $\forall x \in U$: $B(x) > 0$, where U is the set of unsafe states,

(3) $\forall x$: $B(x) = 0 \Rightarrow (\nabla B)^T \bullet f(x) < 0$, where $(\nabla B)^T$ is the transpose of gradient $\nabla B = (\frac{\partial B}{\partial x_1}, \ldots, \frac{\partial B}{\partial x_n})$ and $f(x)$ defines the system dynamics.

Condition (3) ensures future system states are safe by ensuring the separation the set of unsafe states from the set of reachable states from the given initial states X_0. Thus a barrier certificate provides an unbounded-time safety certificate of the system.

The key idea is to find a symbolic representation of a barrier certificate from sample simulation traces. This analysis technique was first used to analyze hybrid systems in [15], and more recently applied to study CPSs with LECs [18]. We have adapted the process in [18] to find candidate barrier certificates using optimization system Pyomo [9] with linear solver Gurobi and to validate a barrier certificate using SMT solver Z3.

First, a candidate barrier certificate W (similar to find a Lyapunov candidate in stability analysis is found using a typical template (sum of squares polynomials): $W(x) = x^T P x$, where $P \in \mathbb{R}^{m \times m}$ is symmetric. The key is to find the values of P using linear constraints: $W(x[t_i]) > 0$ and $W(x[t_i]) - W(x[t_{i+1}]) > 0$, where $x[t_i]$ $(0 \leq i \leq N)$ is a simulation trace of the closed loop (including both plant and DNN controller) system dynamics f. The above linear constraints correspond to the negations of conditions (1) and (3) in barrier certificate. W is a positive function and decreases along system trajectories. Then, a level l is found such that $B(x) = W(x) - l$, where l is a non-negative real number that separates X_0 from U. The overall process in [18] is shown in Fig. 3.

In the above process:

- Equation (3.1): $\forall x \in D.\,(x \notin X_0 \wedge (\nabla W)^T \bullet f(x) \geq -\gamma)$
- Equation (3.2): $\exists x \in X_0.\,(x \notin \{x | W(x) - l \leq 0\})$
- Equation (3.3): $\exists x \in \{x | W(x) - l \leq 0\}.\ (x \in U)$

Equations (3.2) and (3.3) define the opposite of separation, i.e. unsafe.

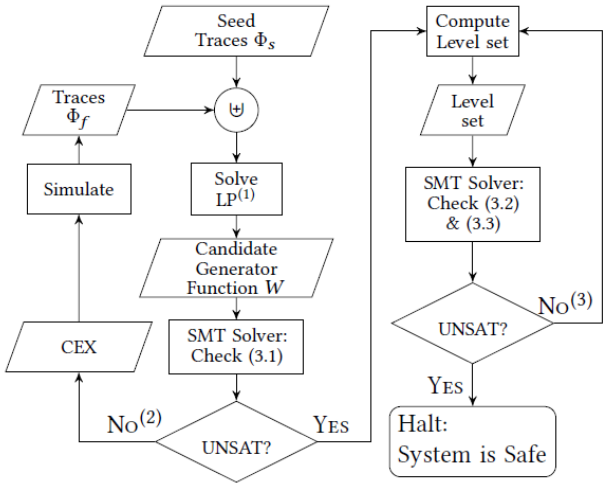


Fig.3. The process of finding barrier certificate

The barrier certificate technique is applied to the car system presented in a previous section. Eight simulation traces (200 steps) of the LEC automated vehicle are run and collected around the region x: [-1,1], y: [49, 51], and θ: [-5, 5]. Python is used to process the raw simulation data into 10 steps of data of system error dynamics (distance error: $\sqrt{x^2+y^2}$-50, where 50 is the radius of the circular track, and angular error: 0). The sum of squares polynomials template is used to fit the simulation data, and the resulting equations are solved using optimization system Pyomo [9] with solvers glpk and Gurobi. Among the 8 sets of equations, four are successfully solved while the other four have no solutions.

One of the candidate barrier certificate is $W(d,\theta) = 0.776 * d^2 + 0.2 * \theta * d + 0.013\theta^2$, where $d = \sqrt{x^2+y^2} - 50$ and $\nabla W = (\frac{\partial W}{\partial d}, \frac{\partial W}{\partial \theta}) = (1.552 * d + 0.2 * \theta,\ 0.2 * d + 0.026 * \theta)$. The error dynamics is $f = [\dot{d}, \dot{\theta}]$: $\dot{d} = (x * \dot{x} + y * \dot{y})/\sqrt{x^2+y^2} = (x * \sin(\theta) + y * \cos(\theta))/\sqrt{x^2+y^2}$, and Equation (3.1) is $\forall x \in D.\,(x \notin X_0 \wedge (\nabla W)^T \bullet f(x) \geq -\gamma)$, where $\gamma = 0.0001$, which is formulated as a constraint satisfaction problem using Pyomo and solved using Z3. Z3 confirms $W(d,\theta)$ as a valid barrier certificate candidate. Z3 is then applied to check Equation (3.2) in finding level $l = 60$, and thus the barrier certificate $B(x,y,\theta) = W(x,y,\theta) - 60$. Finally, Z3 is used to check Equation (3.3) to ensure the safety region, where unsafe region is defined by half spaces: $x \leq -5$, $x \geq 5$, $y \leq 48$, $x \geq 52$.

B. Bounded Model Checking Technique

Bounded model checking was first developed to analyze safety properties of discrete systems [3]. In bounded model checking, the following logic formula ϕ_k is constructed from a given system model and property: $\phi_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} f(s_i)$, where $I(s_0)$ is the characteristic function of the initial state s_0, $T(s_i, s_{i+1})$ is the characteristic function of the transition relation, and $f(s_i)$ represents the negated safety property in unrolled state s_i ($0 \leq i \leq k$). If ϕ_k is satisfiable, there is a transition sequence or a trace from the initial state s_0 to a state s_i that satisfies f, thus violates the safety property. An SMT solver is used to check ϕ_k. Bounded model checking can be used to find violation of a safety property through a counter example, and can ensure a safety property up to k steps.

Bounded model checking techniques have been generalized to analyze hybrid systems with limited successes in the past decade ([2], [4], [5], [13]) and thus can be applied to analyze CPSs. However formal analysis techniques for LECs modeled using DNNs barely exist, a few existing works ([8], [12], [16]) can only handle simple activation functions such as ReLU.

A recent work [11] shows promise to formally analyze CPSs with LECs, in which the LEC modeled in DNN is transformed into a hybrid automaton, and then the overall system is analyzed by composing the LEC generated hybrid automaton with the hybrid automaton modeling the rest of the system. The overall closed loop system model in [11] is shown in Fig. 4.

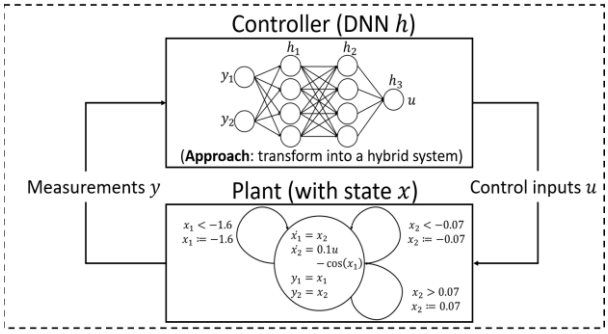


Fig.4. The closed loop system of a controller with plant

The plant dynamics is defined by $\dot{x} = f_p(x,u)$ and $y = g(x)$, where $x \in \mathfrak{R}^n$ is system state, and $u \in \mathfrak{R}^m$ is the input, f_p is a locally Lipschitz-continuous vector field, g: $\mathfrak{R}^n \to \mathfrak{R}^q$. The DNN controller is defined by $u = h(y)$, where h: $\mathfrak{R}^q \to \mathfrak{R}^m$. The closed loop system dynamics: $\dot{x} = f_p(x, h(g(x)))$.

The DNN is transformed into a hybrid automaton as follows: $h(y) = h_L \circ h_{L-1} \circ \ldots \circ h_1(y)$, where each hidden layer i has an element-wise sigmoid activation function $h_i(y) = 1/(1 + e^{-(W_i y + b_i)})$, with the last layer being a linear function: $h_L(y) = W_L y + b_L$. The derivative of sigmoid function $\sigma(y) = 1/(1+e^{-y})$ is $\frac{d\sigma}{dy} = \sigma(y)(1 - \sigma(y))$. The timed proxy function $\sigma(t,y) = 1/(1+e^{-ty})$ is used to define evolution with derivative: $\dot{\sigma}(t,y) = \frac{\partial \sigma}{\partial t} = y\sigma(t,y)(1 - \sigma(t,y))$. The resulting hybrid automaton has one mode for each DNN layer. A set of continuous variables (each corresponding to a neuron) is introduced. The flow of continuous variable y_j in each mode i

is defined by proxy function $\sigma_{ij}(t, y)$ from t = 0.5 to 1 defined using differential equation $\dot{\sigma_{ij}}(t, y)$. Another set of continuous variables are used to keep track the linear functions (weighted sums) of each neuron and have constant change rates 0. Discrete jumps between modes happen when $t = 1$. The overall closed loop system model (composition) is $S = h \,||H_p$, where H_p is the plant automaton with dynamics $f_p(x, u)$. The reachability property on plant with initial state X_0 is defined by $\varphi(X_0) \Rightarrow f(x(t)),$ for $t \geq 0$.

A bounded model checking technique has been developed for CPSs with LECs in this research. The approach in [11] is adapted to translate the component of an HPrTN representing a LEC into a hybrid automaton. The translation of the rest part of HPrTN to a hybrid automaton is straightforward based on the relationships between HPrTNs and hybrid automata given in [6]. Bounded model checker dReach [13] with dReal [5] for non-linear hybrid automata are used to check the resulting composed hybrid automaton.

The bounded model checking technique is applied to the car system presented in a previous section. The Plant (Car) H_p has 3 continuous variables – location *x* and *y* and direction angle θ, and three modes – forward, left, and right. The controller (DNN) h has three layers with one hidden layer containing sigmoid activation function. The overall closed loop system model $S = h \,||H_p$ contains 6 modes (after reduction), and twelve continuous variables (including time). The reachability property to check (the car off the center of circular track of radius 50 by 5) is $sqrt(x^2 + y^2) - 50 > 5 \vee sqrt(x^2 + y^2) - 50 < -5$. The model checking results are shown Table I below:

Table I Bounded model checking results

Model	#Mode	#Depth	#ODEs	#Vars	Precision(δ)	Result	Feasible Paths	Time (sec)
AV-LEC@1	6	10	11	12	0.001	UNSAT	29	0.22
AV-LEC@2	6	10	11	12	0.001	UNSAT	29	0.26
AV-LEC@3	6	10	11	12	0.001	UNSAT	41	0.23
AV-LEC@4	6	10	11	12	0.001	UNSAT	41	0.26
AV-LEC@5	6	10	11	12	0.001	UNSAT	29	0.23
AV-LEC@6	6	10	11	12	0.001	UNSAT	29	0.36

V. CONCLUDING REMARKS

HPrTNs are well suited for modeling CPSs with LECs due to (1) their capability to capture concurrency and hybrid behaviors in typical CPSs, (2) their graphical representation and executability to naturally fit the machine learning techniques based DNNs and RL, and (3) their distributed and concurrent data flow computational model to easily integrate different system components. This paper contributes a unique analysis methodology to analyze CPSs with LECs modeled using HPrTNs. Although both barrier certificate and bounded model checking techniques have been around for decades, their applications to deal with LECs only happened in recent years. Integrating both analysis techniques within the same framework with unique tool support is a new contribution of this work.

More case studies will be carried out to show the effectiveness and scalability of this analysis methodology. Additional research will be done to develop new techniques to analyze the robustness of DNNs and the overall stability and safety of CPSs with LECs built using neural fitted RL.

ACKNOWLEDGMENT

This work was partially supported by AFRL under FA8750-15-2-0106 and FA9550-15-0001. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] R. Alur: "Principles of Cyber-Physical Systems", The MIT Press, 2015.

[2] K. Bae and S. Gao: "Modular SMT-based analysis of nonlinear hybrid systems," 2017 Formal Methods in Computer Aided Design (FMCAD), Vienna, 2017, pp. 180-187.

[3] E. Clarke, A. Biere, R. Raimi, Y. Zhu: "Bounded model checking using satisfiability solving". Formal Methods in System Design 19(1), 7–34 (2001).

[4] A. Cimatti, S. Mover, and S. Tonetta: "SMT-based verification of hybrid systems". In Proc. AAAI, 2012.

[5] S. Gao, S. Kong, and E. M. Clarke: "dReal: An SMT solver for nonlinear theories over the reals". In CADE, volume 7898 of LNCS, pages 208–214. Springer, 2013.

[6] X. He and D. Alam: "Hybrid Predicate Transition Nets - A Formal Method for Modeling and Analyzing Cyber-Physical Systems", Proc. of The 2019 IEEE International Conference on Software Quality, Reliability & Security (QRS'19), Sofia, Bulgaria, 2019, 216-227.

[7] X. He: "Modeling and Analyzing Cyber Physical Systems with Learning Enabled Components using Hybrid Predicate Transition Nets", Proc. of 2021 IEEE 21th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Hainan, China, 2021.

[8] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu: "Safety verification of deep neural networks", In International Conference on Computer Aided Verification, 2017, Springer, 3–29.

[9] W. E. Hart, C. D. Laird, J. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, J. D. Siirola: "Pyomo – Optimization Modeling in Python", Springer, 2017.

[10] R. Hafner, M. Riedmiller: "Reinforcement learning in feedback control – Challenges and benchmarks from technical process control", Mach Learn (2011) 84:137–169.

[11] R. Ivanov, J. Weimer, R. Alur, G. J Pappas, and I. Lee: "Verisig: verifying safety properties of hybrid systems with neural network controllers" In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC'19), 2019, 169–178.

[12] G. Katz, C. Barrett, D. L Dill, K. Julian, and M. J Kochenderfer: "Reluplex: An efficient SMT solver for verifying deep neural networks", In International Conference on Computer Aided Verification. Springer, 2017, 97–117.

[13] S. Kong, S. Gao, W. Chen, and E. M. Clarke: "dReach: □-reachability analysis for hybrid systems". In TACAS, volume 9035 of LNCS. Springer, 2015.

[14] S. Liu, R. Zeng, X. He: "PIPE+ - A Modeling Tool for High Level Petri Nets", Proc. of International Conference on Software Engineering and Knowledge Engineering (SEKE11), Miami, July 2011, 115 - 121.

[15] S. Prajna and A. Jadbabaie: "Safety Verification of Hybrid Systems Using Barrier Certificates". In In Hybrid Systems: Computation and Control. Springer, 477–492, 2004.

[16] L. Pulina and A. Tacchella: "An Abstraction-Refinement Approach to Verification of Artificial Neural Networks". In Proc. 22nd Int. Conf. on Computer Aided Verification (CAV), pages 243-257, 2010.

[17] R. S. Sutton, and A. G. Barto: "Reinforcement Learning: An Introduction", (2nd edition), Cambridge, MA: MIT Press, 2018.

[18] C. Tuncali, J. Kapinski, H. Ito, J. Deshmukh: "Reasoning about Safety of Learning-Enabled Components in Autonomous Cyber-physical Systems". Design Automation Conference (DAC) 2018.

Problem-specific knowledge based artificial bee colony algorithm for the rectangle layout optimization problem in satellite design

Yichun Xu[1,2],Shuzhen Wan[2],Fangmin Dong[2]
[1]*Hubei Province Engineering Technology Research Center for Construction Quality Testing Equipments*
[2]*College of Computer and Information Technology*
China Three Gorges University, Yichang 443002, Hubei, China
xuyichun@ctgu.edu.cn, wanshuzhen@163.com, fmdong@ctgu.edu.cn

***Abstract*—The layout optimization problem is brought from the design of the recoverable satellite, where a set of objects (equipments or devices) are required to be installed on a circular load board. The aim of the problem is to find a layout of the objects with no interference, less unbalance, and less space occupied. Artificial bee colony (ABC) algorithms show good performance in many engineering problems. In this article, based on the analysis of the solution distribution, a problem-specific knowledge based ABC is proposed, which is configured with special initialization and parameter settings. On an open benchmark with ten instances, the proposed ABC is compared with two widely used algorithms. Its performance outperforms the genetic algorithm on all the instances, and outperforms the quasi-human algorithm on nine instances.**

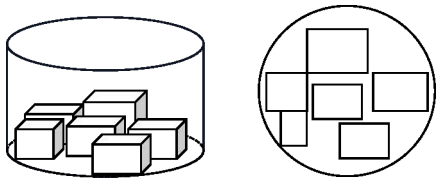
***Keywords*—swarm intelligence; artificial bee colony algorithm; layout optimization problem; weighted rectangle packing**

I. Introduction

In the design of the recoverable satellite, some objects (devices or equipments) are required to be installed on a circular load board (Fig. 1). Three kinds of constrains or objectives should be concerned: 1. There should be no interference between objects. 2. The layout of the objects should be compact so that they occupy less space. 3. The unbalance of the system should be small enough, so the system is easy to control. In this article, we study the two-dimensional problem that the shapes of the objects are modeled as rectangles, which is called the rectangle layout optimization problem (RLOP).

RLOP was first proposed in [1], where the authors studied the isomorphism of the layouts by graph theory and group theory, and then proposed a global optimization framework. From then on, some meta-heuristics were proposed for this problem, such as the genetic algorithms (GA)[2], the particle swarm optimization (PSO) [3, 4], the simulated annealing algorithm (SA) [5], and the ant colony optimization (ACO) [6]. Another class of algorithms are based on the quasi-physics and quasi-human strategies [7], that an elastic potential energy function is defined to measure the overlaps between objects, and then the overlaps are reduced by the elastic force step by step. Recently, [8] proposed a three-dimensional model, that the items have height and there are multiple layers to install the items. Except for assigning a layer for each item, their layout optimization algorithm is basically an application of the algorithm for two-dimensional models. In general, the existing algorithms are with good results on the small-scale RLOP. With the increase of the rectangle numbers, they become time consuming and the solutions decline in quality.

DOI reference number: 10.18293/SEKE2022-014



(a) 3D model (b) 2D model

Fig. 1: Installing equipments on the load board of a recoverable satellite

This article aims at designing a new algorithm for RLOP with the help of some "guiders". The main idea is based on the observation that in a good layout, the bigger and the heavier objects often locate at the center of load board. This observation leads us to design a new greedy strategy. We then combine the greedy strategy into an artificial bee colony (ABC) framework. The artificial bee colony (ABC) algorithm is a kind of nature-inspired optimizer in swarm intelligence proposed by Karaboga [9]. Because ABC algorithm needs fewer parameters and the performance is often good, it is very popular in most engineering fields [10].

On an open benchmark with ten instances, the performance of the proposed ABC outperforms the widely used genetic algorithm [2] and the quasi-human algorithm [7].

II. Mathematical model

In two-dimensional case, we need to pack a set of rectangles with masses into a containing circle as Fig. 1(b). For the constraints, the rectangles cannot overlap each other, and the center of mass of the system should be near the center of the containing circle to keep the equilibrium. The aim of the

problem is to minimize the envelopment circle which covers all the rectangles. The problem can also be stated as follows.

Define n rectangles by a list $R = (l_1, w_1, m_1), (l_2, w_2, m_2), \ldots, (l_n, w_n, m_n)$, where l_i, w_i, and m_i are the length, the width, and mass of rectangle i. Assume the center of mass and shape is located at the same point in each rectangle. In a two-dimensional Cartesian coordinate system, we set the Cartesian origin to the center of the containing circle. The list $X = (x_1, y_1, \theta_1), (x_2, y_2, \theta_2), \ldots, (x_n, y_n, \theta_n)$ denotes a layout, where x_i, y_i is the center of the rectangle i, and θ_i denotes its orientation. The aim of the problem is to find a layout X to satisfy the following constraints:

1) $\theta_i \in \{0, 1\}$, where θ_i=0 or 1 mean that the edge with length l_i is parallel or perpendicular to the x axis, then the items are placed orthogonal to each other.
2) There is no overlap between any two rectangles, that is, for all $i \neq j$, at least one of the following conditions should be satisfied:

$$x_i + l'_i/2 \leq x_j - l'_j/2 \quad (1)$$
$$x_i - l'_i/2 \geq x_j + l'_j/2 \quad (2)$$
$$y_i + w'_i/2 \leq y_j - w'_j/2 \quad (3)$$
$$y_i - w'_i/2 \geq y_j + w'_j/2 \quad (4)$$

where l'_i and w'_i are related to the length and width after considering the orientation θ_i, which satisfy

$$l'_i = l_i(1 - \theta_i) + w_i\theta_i \quad (5)$$
$$w'_i = w_i(1 - \theta_i) + w_i\theta_i \quad (6)$$

3) The center of mass of all rectangles should be located at the center of the circle for equilibrium, that is, given a small positive permissible value of δ

$$(x_w, y_w) = \left(\frac{\sum_{i=1}^{i=n} m_i x_i}{\sum_{i=1}^{i=n} m_i}, \frac{\sum_{i=1}^{i=n} m_i y_i}{\sum_{i=1}^{i=n} m_i}\right) \quad (7)$$
$$\sqrt{x_w^2 + y_w^2} \leq \delta \quad (8)$$

such that the radius r of the envelopment circle is minimized, where

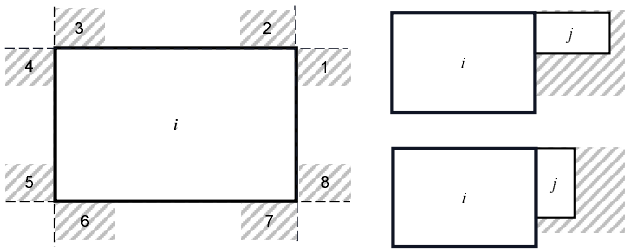
$$r = \max_{1 \leq i \leq n} \left(\sqrt{(|x_i| + l'_i/2)^2 + (|y_i| + w'_i/2)^2} \right). \quad (9)$$

III. ABC ALGORITHM

Before introducing the design of ABC for RLOP, we first give out a constructive heuristic to compose a layout, which is an important building block.

A. Constructive kernel heuristic (CKH)

The ABC algorithm is based on a constructive heuristic first appeared in [2]. At first, all the rectangles are waited in a queue, and the first rectangle is packed in the center of the circle. When packing a rectangle, for the goal of minimizing envelopment radius, we require it close to an already packed rectangle. As in Fig. 2(a), an already packed rectangle i provides 8 regions along its edges and vertices. A rectangle j to be packed should choose a region with an orientation



(a) regions (b) placement

Fig. 2: The regions provided by a rectangle(a) and the placement of a rectangle in a region with different orientation(b)

(Fig. 2(b)), so there are totally 16 schemes to place j beside i. After deletion of the schemes with overlap, there are still many feasible schemes besides all the packed rectangles, and the algorithm will select one by a greedy strategy–the region leads to the minimal temporary envelopment circle will be chosen.

B. Inversion count and distribution of the solutions

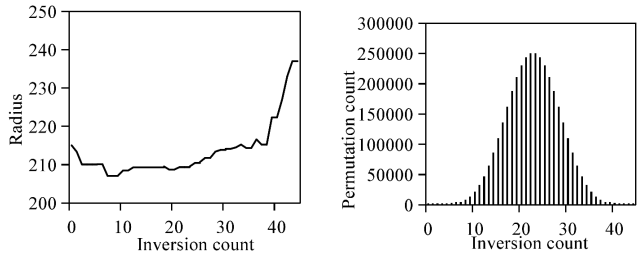
The application of CKH in this article is different from [2]. In the CKH, the output layout X is dependent on the permutation p. Because there are $n!$ permutations in total, the blind search such as [2] has very low efficiency. According to the computational practice, we find that the bigger and heavier objects should be placed first, which means a greedy permutation $\boldsymbol{g}$ in the descending order of $l_i w_i m_i$ can lead to a good layout. This finding relates the concept of inversion count to the goodness of a permutation.

For a permutation p, an inversion exists between the items p_i and p_j, if $l_{p_i} w_{p_i} m_{p_i} < l_{p_j} w_{p_j} m_{p_j}$ and $i < j$. According to the number theory, the inversion count of a permutation ranges from 0 to $\frac{n(n-1)}{2}$, and the greedy permutation g has a inversion count of 0.

By the computational experiences, a permutation with small inversion count often results in a better layout. On a randomly chosen RLOP instance with 10 rectangles (larger instance is in similar situation), we enumerate all the permutations and get the corresponding layouts and their envelopment radii by CKH. The minimal radii from the permutations with same inversion count are plotted in Fig. 3(a). The number of permutations with the same inversion count are illustrated in the histogram Fig. 3(b). We found that the permutations with inversion count less than 10 are obvious have better results, Moreover, the number of such types of permutations is relatively small that it is easy to search them.

C. ABC algorithm based on inversion count

ABC algorithm is a meta-heuristic based on the foraging behavior of honey bees. There are three kinds of bees in a colony. The employed bees work on a food source and share the information with the onlooker bees by dancing. The onlooker bees select a food source after watching the dance and try to improve it. When the food source is exhausted, the



(a) minimal radius (b) histogram

Fig. 3: The distribution of the solutions

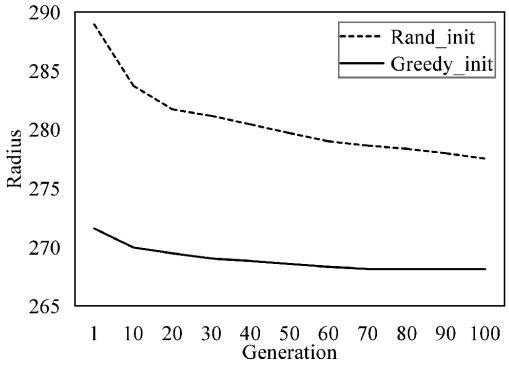


Fig. 4: Results on different initialization

related employed bee becomes a scout bee, who tries to find a new resource. The ABC algorithm for the RLOP is described in algorithm 1.

Algorithm 1 ABC algorithm

```
Initialize the first generation s_1, s_2, ..., s_n by the greedy strategy
for generation = 2 to G do
    for i=1 to n do {Employed phase}
        s_i:=best of (s_i, INSERT(s_i))
    end for
    for i=1 to n do {Onlooker phase}
        Randomly select a s_k with probability of P(s_k)
        s_k:=best of (s_k, INSERT(s_k))
    end for
    for i=1 to n do {Scout phase}
        if s_i is not improved for T generations then
            Restore s_i from the first generation
        end if
    end for
end for
return the best solution
```

1) Individual and fitness function: The individual solution s_i is a permutation p, which can be evaluated by the fitness function as (10), where $r(s_i)$ is the envelopment radius result of CKH.

$$f(s_i) = \frac{1}{r(s_i)} \tag{10}$$

2) first generation: The first generation defines the start points of the search. Based on the analysis in section III-B, we should focus on the permutations with small inversion count. We initial the fist generation with the greedy permutation g and other $n-1$ permutations generated by swapping the adjacent elements of g. The inversion count of the first generation is less than or equal to 1.

3) Mutation operator : In the employed phase, we choose the INSERT mutation operator like the genetic algorithm [11]. In a permutation p, after a block of $p_i, p_{i+1}, \ldots, p_{i+k-1}$ is chosen, the INSERT operator moves p_{i+k-1} before p_i. The mutation operator can change the inversion count of the permutation. Moreover, the larger the block size k, the more the inversion count is changed. So the block size k should have an upper bound.

4) Selection probability: The selection probability for an individual in the onlooker phase is proportional to its fitness value, which is defined as (11)

$$P(s_i) = \frac{f(s_i)}{\sum_{i=1}^{n} f(s_i)} \tag{11}$$

IV. Computational results

The numerical experiments were on a Dell OptiPlex 7080 Tower, with a 3.10 GHz Intel i5-10500 CPU,16 GB RAM, Win10 OS. And the programs were compiled by Dev C++ 5.9.2. By the experiences, the block size of INSERT is set to 5, and the initialization interval T in the scout phase is set to 10.

A. Experiment 1: Comparison of greedy initialization and random initialization

This experiment is to compare the greedy initialization and the popular random initialization in ABC. 30 instances with 20 rectangles for each were randomly generated that w_i, l_i are in range of [1,200] and m_i is near $w_i l_i$ for each rectangle i. In the experiment, the average radii of the 30 instances were recorded in each generation. The convergence curves are provided in Fig. 4. The results show that the greedy initialization is more advantage than the random initialization. Even after 100 generations of search, the average radii with random initialization is still worse than the start point of the greedy initialization. The ABC algorithm with random initialization wastes too much search energy in the subspace of permutation with larger inversion count.

B. Experiment 2: Numerical computation on an open benchmark with large-scale instances

In this experiment, we ran the ABC algorithm on a benchmark provided in [7], that there are 10 test instances R1, R2, ..., R10, and the numbers of rectangles are 10, 20, ..., 100 respectively. Two algorithms in the literatures were selected as the baselines, which are the quasi-human algorithm (IBF) in [7] and the genetic algorithm (GA) in [2].

TABLE I: Results on 10 instances

inst	rtarget	GA		IBF		ABC	
		Fail	time(s)	Fail	time(s)	Fail	time(s)
R1	36.90	5	/	0	113.79	5	/
R2	65.30	5	/	4	746.96	0	378.16
R3	56.05	5	/	5	/	0	530.71
R4	75.30	5	/	5	/	0	50.48
R5	91.55	5	/	5	/	1	317.02
R6	101.79	5	/	5	/	0	88.18
R7	102.89	5	/	5	/	0	128.74
R8	109.09	5	/	5	/	0	407.82
R9	115.32	5	/	5	/	0	1628.23
R10	124.10	5	/	5	/	1	881.53

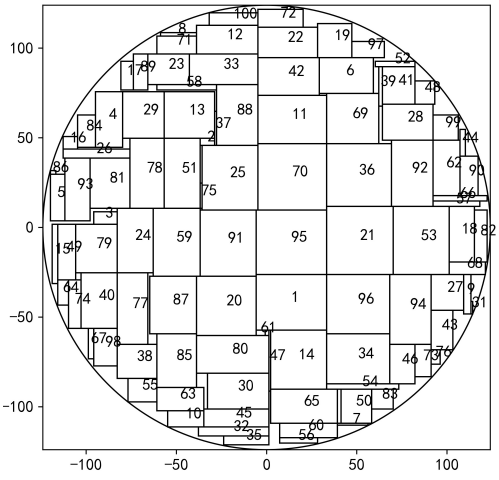
IBF uses the container radius as an input and its objective is to find a layout smaller than the given radius. To make a fair comparison with IBF, we defined a target container radius (**rtarget**) for each instance, and then used the time of finding a valid layout as the metric of performance. We set the stopping criterion of all the three algorithm as obtaining a layout with envelopment radius less than the target, or execution time exceeding an hour. The other parameters of the baselines were set as the literatures. In the ABC, the number of individual n was set to the number of rectangles in each instance. The three algorithms were executed on each instance for 5 times. If an execution did not output a layout in an hour, we marked it as one '**failure**'. Excluding the failed executions, we provide the average time of the rest executions on each instance. The results are listed in Table I and we give the layout diagram of the instance R10 by ABC in Fig. 5.

From the detailed data in Table I, GA fails in all the tests and shows the worst performance among the three algorithms. The reason why GA loses is that it tries to search the whole solution space and wastes much energy on the space with poor solutions, while the proposed ABC makes the search around the greedy solution, many good results are in this subspace.

IBF only passes the first 2 smaller instances, that it gets best results in R1, and gets a layout for R2 after 4 failures. It fails all the tests on the last 8 instances. But on the other side, ABC gets the best results in the rest 9 instances except R1, and passes 43 tests among the total 45 tests. The only two failures in tests of instance 5 and 10 should be because we set harder targets. ABC's failures in R1 is because of the shortcoming of CKH. It places the objects at certain positions, which restricts the pattern of the solution, so it may miss the optimal layout of smaller instances.

V. Conclusion

A problem-specific knowledge based artificial bee colony (ABC) algorithm for the layout optimization problem in the satellite design is presented in this article. After the investigate of the distribution of the solutions, the ABC algorithm is designed to search the most likely subspace containing high quality solutions, so that it can easily find a good solution in short time. On an open benchmark with 10 instances, ABC algorithm is compared with two widely used algorithms. It outperforms the genetic algorithm on all the instances, and outperforms the quasi-human algorithm on nine of them. The proposed ABC algorithm may have great practical value to find the rational layout of the objects in the aerospace industry.



(a) R10

Fig. 5: Layout diagram of ABC on R10

References

[1] E. Feng, X. Wang, X. Wang, and H. Teng, "A global optimization agorithm for layout problems with behavior constraints," *Applied Mathematics, A Journal of Chinese Universities*, vol. 14, no. 1, pp. 98–104, 1999.

[2] Y. Xu, F. Dong, Y. Liu, and R. Xiao, "Genetic algorithm for rectangle layout optimization with equilibrium constraints," *Pattern Recognition and Artificial Intelligence*, vol. 23, no. 6, pp. 794–801, 2010.

[3] Y.-C. Xu, R.-B. Xiao, and M. Amos, "Particle swarm algorithm for weighted rectangle placement," in *the 3rd Int'l Conf. on Natural Computation*, pp. 728–732, 2007.

[4] Z. Huang and R. Xiao, "Hybrid algorithm for the rectangular packing problem with constraints of equilibrium," *Journal of Huazhong University of Science and Technology (Natural Science Edition)*, vol. 9, no. 3, pp. 96–99, 2011.

[5] Y.-C. Xu, R.-B. Xiao, and M. Amos, "Simulated annealing for weighted polygon packing." https://arxiv.org/abs/0809.5005, 2008.

[6] M. Ji and R. Xiao, "Ant colony optimization and heuristic algorithms for rectangle layout optimization problem with equilibrium constraints," *Journal of Computer Applications*, vol. 30, no. 11, pp. 2898–2901, 2010.

[7] J. Liu, J. Li, Z. Lv, and Y. Xue, "A quasi-human strategy-based improved basin filling algorithm for the orthogonal rectangular packing problem with mass balance constraint," *Computers and Industrial Engineering*, vol. 107, pp. 196–210, 2017.

[8] C.-Q. Zhong, Z.-Z. Xu, and H.-F. Teng, "Multi-module satellite component assignment and layout optimization," *Applied Soft Computing*, vol. 75, pp. 148–161, 2019.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. tr062005, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[10] B. Akay, D. Karaboga, B. Gorkemli, and E. Kaya, "A survey on the artificial bee colony algorithm variants for binary, integer and mixed integer programming problems," *Applied Soft Computing*, vol. 106, no. 3, p. 107351, 2021.

[11] M. Serpell and J. Smith, "Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms," *Evolutionary Computation*, 2010.

Modeling and Verifying AUPS Using CSP

Hongqin Zhang, Huibiao Zhu*, Jiaqi Yin, Ningning Chen
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China

***Abstract*—The Internet of Things (IoT) is an important technology in IT industries. The wide adoption of IoT raises concerns about security and privacy. The Authenticated Publish/Subscribe (AUPS) model is an IoT system which aims to address the security and privacy issues in the IoT environment. AUPS is attracting more and more attention from industries. Hence, the reliability of AUPS is worth investigating.**

In this paper, we model AUPS using Communicating Sequential Processes (CSP). Five properties (Deadlock Freedom, Data Availability, Data Leakage, Device Faking and User Privacy Leakage) of the model are verified by utilizing the model checker Process Analysis Toolkit (PAT). The verification results demonstrate that AUPS cannot ensure the security of critical data. To solve the problem, we improve the model by using a digital certificate. The verification results of the improved model indicate that our study can enhance the security and reliability of the AUPS model.

***Index Terms*—AUPS, CSP, PAT, Modeling, Verifying**

I. INTRODUCTION

As an important paradigm in IT industries, the Internet of Things (IoT) connects heterogeneous devices to provide users with required services [1]. Communication efficiency, data security and user privacy are the three major issues of IoT [2]. In order to cope with these issues, several solutions have been proposed [3]–[5]. Shi et al. came up with an IoT system [3] which used a machine learning method to improve the efficiency of data processing. Jung et al. proposed a distributed IoT scheme where two or more servers created a group and a client viewed the group as a powerful server [4]. This scheme reduced network latency. However, it cannot protect user privacy. Shang et al. presented a publish/subscribe IoT framework [5]. It adopted data authentication to improve the communication security without considering user privacy issues. These solutions improved communication efficiency. However, they did not fully explore security and privacy issues.

Thus, a publish/subscribe IoT system called Authenticated Publish/Subscribe (AUPS) [6] was proposed. AUPS adopted Attribute-based Access Control (ABAC) [7] to improve the security and privacy of the system. ABAC controls access to data by evaluating rules against the attributes of users. According to the experimental results [6], AUPS was more efficient than the other existing secure solutions. As AUPS is attracting more and more attention from industries, we believe that it is valuable to analyse the functional and security properties of AUPS using formal methods.

In this paper, AUPS is formally modeled using the process algebra CSP [8]. The model checking tool PAT [9] is adopted to verify its functional and security properties. According to the verification results, AUPS may cause data leakage, device faking and user privacy leakage once intruders appear. Thus, we improve the model by using a digital certificate. Then we verify the improved model using PAT. The verification results show that our work can guarantee the security of AUPS.

The rest of this paper is organized as follows. Section II briefly introduces AUPS and CSP. Section III is devoted to the modeling of AUPS. In Section IV, we analyse the verification results and give the improvement that can address the vulnerabilities of the model. Finally, conclusion and future work are given in Section V.

II. BACKGROUND

In this section, we give a brief description of AUPS. After that, we introduce the process algebra CSP.

A. AUPS

As shown in Fig. 1, the Authenticated Publish/Subscribe (AUPS) contains the following entities:

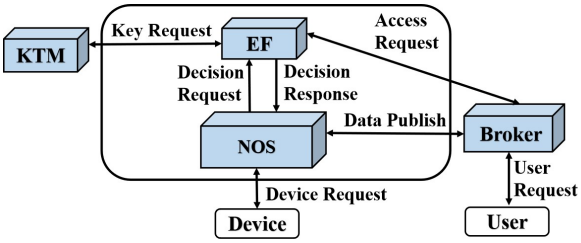


Fig. 1: AUPS schema

- **Device:** It collects data from the environment and sends them to the Networked Smart Object (NOS).
- **Networked Smart Object (NOS):** The NOS deals with the collected data and publishes them to the broker.
- **User:** The user subscribes to services to get the required data by interacting with the broker.
- **Broker:** It handles the subscription requests from users, and then forwards data to the legal subscribers.
- **Enforcement Framework (EF):** The EF is a policy enforcement framework. It performs access control and manages the data encryption/decryption keys using a key table. The structure of key table is shown in TABLE I.
- **Keys Topics Manager (KTM):** It generates the data encryption/decryption keys according to the services.

The core functions of AUPS are data publishing and service subscription. Data publishing represents that the devices publish collected data to the Internet. Service subscription means

*Corresponding author: hbzhu@sei.ecnu.edu.cn (H. Zhu).

DOI reference number: 10.18293/SEKE2022-088

TABLE I: Key table

Field	Description
id	The identifier of the corresponding key
key	The actual key
val	The expiration date of the key
atb	The attribute(s) owned by users allowed to get the data

that users subscribe to services to obtain the required data. We introduce the actions of the two functions as follows. The related notations and descriptions are listed in TABLE II.

TABLE II: Notations and descriptions

Notation	Description
$puk_x/prk_x/sk_x$	Public/Private/Symmetric key of the user/device/ broker/NOS/intruder, $x \in \{u, d, b, n, i\}$
id_x/sid_x	Identity information/Session identifier of the user/ device, $x \in \{u, d\}$

Before publishing data, the device must finish device registration. The related actions are shown in Fig. 2.

- $a1$: A device sends a registration request req_d to the NOS.
- $a2$: When receiving the request, the NOS sends its public key puk_n to the device.
- $a3$: The device sends its identity information id_d and symmetric key sk_d encrypted with puk_n to the NOS.
- $a4$: The NOS decrypts the message to get id_d and sk_d using its private key prk_n, and then distributes a session identifier sid_d encrypted with sk_d to the device.
- $a5$: The device sends the collected data d and session identifier sid_d encrypted with sk_d to the NOS.
- $a6$: The NOS decrypts the message to get the data d using sk_d, and then asks the EF for a data encryption key.
- $a7$: The EF sends a key kT to the NOS. After that, the NOS can publish data encrypted with kT to the broker.

Before getting data, the user needs to register and subscribe to relevant services. The related actions are given in Fig. 3.

- $b1$: A user sends a registration request req_u to the broker.
- $b2$: The broker sends its public key puk_b to the user.
- $b3$: The user sends its private information id_u and symmetric key sk_u encrypted with puk_b to the broker.
- $b4$: The broker decrypts the message to get id_u and sk_u using prk_b, and then sends the session identifier sid_u and attribute at to the user. Notice that sid_u is used to identify the user without revealing its sensitive information.
- $b5$: The user sends a subscription request $reqT$ encrypted with sk_u to the broker.
- $b6$: The broker verifies the identity of the user, and then sends an access control request to the EF.
- $b7$: The EF checks whether the user can access the data of service T. If the result is positive, the EF forwards the data decryption key kT to the broker.
- $b8$: The broker sends kT encrypted with sk_u to the user. Finally, the user can obtain the data using kT.

B. CSP

Communicating Sequential Processes (CSP) is a process algebra proposed by C. A. R. Hoare [8]. Here we briefly introduce the syntax of CSP used in this paper.

$$P, Q ::= a \to P \mid c?x \to P \mid c!v \to P \mid P; Q \mid P||Q \mid P \square Q \mid P \lhd b \rhd Q \mid P[[a \leftarrow b]]$$

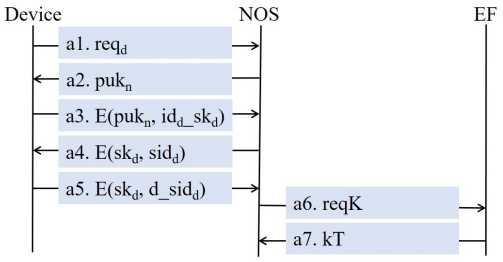


Fig. 2: Data publishing

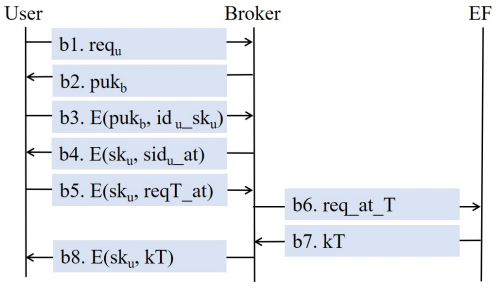


Fig. 3: Service subscription

- $a \to P$ indicates that a process performs action a first, and then acts like process P.
- $c?x \to P$ represents that a process receives a message via channel c and assigns the received message to x, and then behaves like process P.
- $c!v \to P$ denotes that message v is sent through channel c, and then process P is executed.
- $P; Q$ is the sequential execution of processes P and Q.
- $P||Q$ describes that processes P and Q run in parallel.
- $P \square Q$ stands for the general choice of processes P and Q, and the selection is made by the environment.
- $P \lhd b \rhd Q$ shows that if the condition b is true, process P is executed, otherwise process Q is executed.
- $P[[a \leftarrow b]]$ means renaming action. Event a in process P is replaced by event b.

III. MODELING

In this section, we focus on the modeling of AUPS. First, we introduce some preparatory notations for the modeling including sets, messages and channels. Based on these notations, we give the formal model of AUPS.

A. Sets, Messages and Channels

First, we explain the sets defined in our model. **Entity** is a set of entities described in Section II. **Req** set contains requests of entities. **Key** set is composed of all the keys. **Data** set contains the data collected by devices. **Inf** set denotes other message contents including identifier set ID, feedback message set Ack, attribute set Atb and service set $Service$.

Besides, we define the encryption function E and decryption function D to model the messages:

E(k, m); D(k, E(k, m)); D(k^{-1}, E(k,m))

Function $E(k, m)$ means that we encrypt the message m using k. $D(k, E(k, m))$ denotes that we use a symmetric key

k to decrypt the message which is encrypted by k. D(k^{-1}, E(k,m)) indicates that we use the corresponding decryption key k^{-1} to decrypt the message encrypted by k.

Based on the sets and functions defined above, we abstract and classify the messages as follows:

$$MSG_{req} = \{msg_{req}.a.b.req, msg_{req}.a.b.E(k, req) \mid a, b \in Entity, k \in Key, req \in Req\}$$
$$MSG_{key} = \{msg_{key}.a.b.E(k_1, k) \mid a, b \in Entity, k_1, k \in Key\}$$
$$MSG_{inf} = \{msg_{inf}.a.b.inf \mid a, b \in Entity, inf \in Inf\}$$
$$MSG_{data} = \{msg_{data}.a.b.d, msg_{data}.a.b.E(k, d) \mid a, b \in Entity, d \in Data, k \in Key\}$$
$$MSG_{in} = \{msg_{req1}.t, msg_{key1}.k \mid t \in Service, k \in Key\}$$
$$MSG_{out} = MSG_{req} \cup MSG_{inf} \cup MSG_{key} \cup MSG_{data}$$
$$MSG = MSG_{out} \cup MSG_{in}$$

MSG_{req} represents the set of request messages. MSG_{key} denotes the set of messages containing the keys. MSG_{inf} involves messages containing identifiers, feedback messages, attributes and services. MSG_{data} consists of messages containing the data collected by devices. MSG_{out} means the set of messages transmitted between entities. MSG_{in} consists of the internal processing messages of entities. MSG is composed of all the messages in the model.

Then we give the definitions of communication channels:

- Channels between honest entities shown by ***COM_PATH***: $ComDN,\ ComNB,\ ComUB,\ ComBE,\ ComNE,\ GetE,\ ComEK$
- Channels for intruders to intercept or fake the transmitted messages denoted by ***INTRUDER_ PATH***: $FakeDN,\ FakeND,\ FakeUB,\ FakeBU$

The declaration of the channels is given as follows:

$$Channel\ COM\_PATH, INTRUDER\_PATH : MSG$$

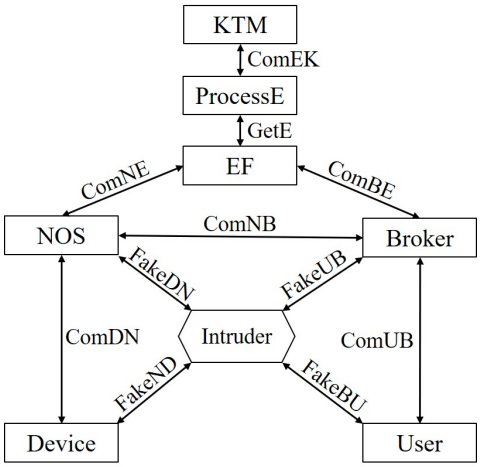


Fig. 4: Channels of AUPS model

B. Overall Modeling

In this subsection, we give the whole model of AUPS. $System_0$ only contains legal entities which are running in parallel. Based on the model $System_0$, we construct the model of $System$ by introducing attacks from intruders.

$$System_0 =_{df} Broker \| User \| EF \| ProcessE \| KTM \| Device \| NOS \| Clock$$
$$System =_{df} System_0 [| INTRUDER\_PATH |] Intruder$$

$Broker$, $User$, EF, $Device$, NOS and KTM are processes describing the behavior of the broker, user, EF, device, NOS and KTM respectively. $ProcessE$ denotes the internal processing procedures of the EF. The process $Clock$ is used to realize the synchronization of time. In addition, the process $Intruder$ simulates the actions of intruders. The channels between all processes are shown in Fig. 4.

C. Clock Modeling

AUPS adopts a temporary key to encrypt the published data. Before using the temporary key, the entity needs to check the expiration date of the key. Hence, we define the process $Clock$ to realize the synchronization of all entities. The process $Clock$ serves to record the time and return the current time whenever the entities want it.

$$Clock(t) =_{df} tick \rightarrow Clock(t+1) \ \square \ Time!t \rightarrow Clock(t)$$

D. User Modeling

We formalize the process $User_0$ to describe the behavior of the user without intruders as follows:

$$\begin{aligned} User_0 =_{df}\ & ComUB!msg_{req}.U.B.req_u \rightarrow \\ & ComUB?msg_{key}.B.U.puk_b \rightarrow \\ & ComUB!msg_{req}.U.B.E(puk_b, id_u.sk_u) \rightarrow \\ & ComUB?msg_{inf}.B.U.E(sk_u, sid_u.at) \rightarrow \\ & \left(\begin{array}{l} ComUB!msg_{req}.U.B.E(sk_u, reqT.at) \rightarrow \\ ComUB?msg_{key}.B.U.E(sk_u, kT) \rightarrow \\ \left(\begin{array}{l} ComUB?msg_{data}.B.U.E(kT, d) \rightarrow \\ User_0 \lhd D(kT, E(kT, d)) \rhd (fail \rightarrow User_0) \end{array} \right) \\ \lhd D(sk_u, E(sk_u, kT)) \rhd (fail \rightarrow User_0) \end{array} \right) \\ & \lhd D(sk_u, E(sk_u, sid_u.at)) \rhd (fail \rightarrow User_0) \end{aligned}$$

The above actions correspond to $b1 - b5$ and $b8$ in Fig. 3. First, the user sends a registration request req_u to the broker and receives the broker's public key puk_b. Then the user requests an attribute by sending its identity information id_u and symmetric key sk_u encrypted with puk_b to the broker. Once getting the attribute at and session identifier sid_u, the user sends a subscription request $reqT$ encrypted with sk_u to the broker. If the request is accepted, the user receives the data decryption key kT and encrypted data. Finally, the user can obtain the required data d using kT. Then we consider attacks from intruders.

Based on the achieved model $User_0$, we formalize the process $User$ with intruders via renaming as follows:

$$\begin{aligned} User =_{df} & User_0[[\\ & ComUB!\{|ComUB|\} \leftarrow ComUB!\{|ComUB|\}, \\ & ComUB!\{|ComUB|\} \leftarrow FakeUB!\{|ComUB|\}, \\ & ComUB?\{|ComUB|\} \leftarrow ComUB?\{|ComUB|\}, \\ & ComUB?\{|ComUB|\} \leftarrow FakeBU?\{|ComUB|\}]] \end{aligned}$$

$\{|ComUB|\}$ represents the set of all communications over the channel $ComUB$. The first two lines mean that whenever $User_0$ transmits a message on the channel $ComUB$, $User$

can transmit the same message on the channel $ComUB$ or $FakeUB$. The last two lines are similar.

E. Broker Modeling

We give the model of process $Broker_0$ to describe the behavior of the broker without intruders as follows:

$$
\begin{array}{l}
Broker_0 =_{df} ComUB?msg_{req}.U.B.req_u \rightarrow \\
\quad ComUB!msg_{key}.B.U.puk_b \rightarrow \\
\quad ComUB?msg_{req}.U.B.E(puk_b, id_u.sk_u) \rightarrow \\
\left(\begin{array}{l}
ComUB!msg_{inf}.B.U.E(sk_u, sid_u.at) \rightarrow \\
ComUB?msg_{req}.U.B.E(sk_u, reqT.at) \rightarrow \\
\left(\begin{array}{l}
ComBE!msg_{req}.B.E.req.at.T \rightarrow \\
ComBE?msg_{inf}.E.B.ack.kT \rightarrow \\
\left(\begin{array}{l}
ComUB!msg_{key}.B.U.E(sk_u, kT) \rightarrow \\
ComNB?msg_{data}.N.B.E(kT, d) \rightarrow \\
ComUB!msg_{data}.B.U.E(kT, d) \rightarrow \\
Broker_0 \lhd (ack == true) \rhd \\
(fail \rightarrow Broker_0)
\end{array} \right) \\
\lhd D(sk_u, E(sk_u, reqT.at)) \rhd (fail \rightarrow Broker_0)
\end{array} \right) \\
\lhd D(prk_b, E(puk_b, id_u.sk_u)) \rhd (fail \rightarrow Broker_0)
\end{array} \right)
\end{array}
$$

The above actions correspond to $b1 - b8$ in Fig. 3. During user registration, the broker sends its public key puk_b to the user, and then obtains the user's identity information id_u and symmetric key sk_u through decryption. After that, the broker distributes the attribute at and session identifier sid_u encrypted with sk_u to the user. When receiving the subscription request $reqT$, the broker verifies the user's identity by sending the attribute at and service T to the EF. If the feedback message ack from the EF is true, the broker sends the data decryption key kT encrypted with sk_u to the user. Otherwise, the subscription request is rejected.

Based on $Broker_0$, the model of the process $Broker$ with intruders can be acquired via renaming similar to the process $User$. We leave out the details.

F. EF Modeling

The EF interacts with the broker and NOS to perform access control. We model the process EF using general choice $\Box$.

$$
\begin{array}{l}
EF =_{df} ComBE?msg_{req}.E.B.req.at.T \rightarrow \\
\quad ack := check(at, T) \rightarrow \\
\left(\begin{array}{l}
GetE!msg_{req1}.T \rightarrow GetE?msg_{key1}.kT \rightarrow \\
ComBE!msg_{key}.E.B.ack.kT \rightarrow EF_0 \\
\lhd (ack == true) \rhd (fail \rightarrow EF_0)
\end{array} \right) \\
\Box ComNE?msg_{req}.N.E.reqK \rightarrow GetE!msg_{req1}.T \rightarrow \\
\quad GetE?msg_{key1}.kT \rightarrow ComNE!msg_{key}.E.N.kT \rightarrow EF_0
\end{array}
$$

The model before $\Box$ describes the communication between the EF and broker. $check(at, T)$ is a function used to verify whether the user with attribute at can access service T. After receiving the broker's request req, the EF adopts $check(at, T)$ to verify the access authority of the user. If the result ack is true, the EF sends a key request to its internal process $ProcessE$. Once receiving the data decryption key kT from $ProcessE$, the EF forwards kT to the broker. These actions correspond to $b6$ and $b7$ in Fig. 3. The model after $\Box$ describes the communication between the EF and NOS. When receiving the NOS's request $reqK$, the EF requests a data encryption key from $ProcessE$, and then forwards the key to the NOS. The related actions are illustrated by $a6$ and $a7$ in Fig. 2.

G. ProcessE Modeling

In order to simulate the internal process of EF, we model $ProcessE$. It mainly deals with the key requests of entities.

$$
\begin{array}{l}
ProcessE =_{df} GetE?msg_{req1}.T \rightarrow kT := findKey(T) \rightarrow \\
\left(\begin{array}{l}
GetE!msg_{key1}.kT \rightarrow ProcessE \\
\lhd (\exists e \in table_i \bullet e.key == kT \wedge Time?t \rightarrow e.val > t) \rhd \\
ComEK!msg_{req}.E.K.reqEK.T \rightarrow ComEK?msg_{key}. \\
K.E.kT \rightarrow GetE!msg_{key1}.kT \rightarrow ProcessE
\end{array} \right)
\end{array}
$$

$findKey(T)$ is a function designed to find the symmetric key that can encrypt or decrypt the data of service T. After receiving the key request from the EF, $ProcessE$ adopts $findKey(T)$ to find a symmetric key kT. Then $ProcessE$ verifies whether kT is valid by checking the expiration date val. If val is later than the current system time, it means that kT has not expired. Then $ProcessE$ sends kT to the EF. If kT has expired, $ProcessE$ requests a new key from the KTM, and then forwards the new key to the EF.

Similarly, we can define CSP processes representing the device, NOS and KTM. The actions of them are introduced in Section II. We omit the details of these processes here.

H. Intruder Modeling

In order to simulate the attacks from the real environment, we model the $Intruder$ process. It can intercept and fake the messages on channel $ComDN$, $ComNB$ and $ComUB$.

First, we define the set of facts that the intruder can learn.

$$Fact =_{df} Entity \cup MSG_{out} \cup \{sk_i, puk_i, prk_i\}$$

Through the known facts, the intruder can deduce new facts. The symbol $F \mapsto f$ means that the intruder can deduce a fact f from the fact set F.

$$
\begin{array}{l}
\{k, c\} \mapsto E(k, c) \\
\{k^{-1}, E(k, c)\} \mapsto c, \quad \{sk, E(sk, c)\} \mapsto c \\
F \mapsto f \wedge F \subseteq F' \Longrightarrow F' \mapsto f
\end{array}
$$

The first rule means encryption. The second and third rules denote the decryption in asymmetric and symmetric encryption forms respectively. The last rule shows that if the fact f can be derived from a fact set F, and F is a subset of F', then the intruder can also deduce f from the larger set F'.

Moreover, we use a function $Info(m)$ to imply the facts that the intruder can learn through intercepted messages.

$$
\begin{array}{l}
Info(msg_{key}.a.b.E(k_1, k)) =_{df} \{a, b, E(k_1, k)\} \\
Info(msg_{data}.a.b.E(k, d)) =_{df} \{a, b, E(k, d)\}
\end{array}
$$

Besides, we introduce a channel $DEDUCE$ for the intruder to deduce new facts. Its definition is given as below:

$$\text{Channel } DEDUCE : Fact.P(Fact)$$

Then the process $Intruder_0$ can be modeled as follows:

$$
\begin{array}{l}
Intruder_0(F) \\
=_{df} \Box_{m \in MSG_{out}} Fake.m \rightarrow Intruder_0(F \cup Info(m)) \\
\quad \Box\Box_{f \in Fact, f \notin F, F \mapsto f} Init\{dl = false\} \rightarrow Deduce.f.F \\
\quad \rightarrow \left(\begin{array}{l}
(\ dl := true \rightarrow Intruder_0(F \cup \{f\})\ ) \\
\lhd (f == d) \rhd \\
(\ dl := false \rightarrow Intruder_0(F \cup \{f\})\ )
\end{array} \right)
\end{array}
$$

When intercepting a message m, the intruder adds $Info(m)$ to its knowledge. If the intruder can decrypt m, it can falsify m and send the modified message to the original receiver.

If the receiver does not recognize that the message has been modified, it means that the intruder successfully fakes as the original sender. Furthermore, the intruder can deduce new facts from its knowledge via the channel $DEDUCE$ and add them to its knowledge. Once the intruder deduces the published data successfully, data leakage occurs. id_u represents the identity information of the user, such as name and address. If the intruder deduces the user's sensitive information id_u, user privacy leakage happens. Now we give the model of $Intruder$. The parameter IK is the initial knowledge of the intruder.

$$Intruder =_{df} Intruder_0(IK)$$
$$where,\ IK =_{df} Entity \cup \{sk_i, puk_i, prk_i\}$$

IV. VERIFICATION AND IMPROVEMENT

In this section, we verify several functional and security properties of AUPS. Based on the verification results and analysis of attacks, we improve the original model and give the new verification results of the improved model.

A. Properties Verification

We use Linear Temporal Logic (LTL) formulas to describe five properties of AUPS. $System()$ denotes the model with intruders. By using the assertion $\#assert\ System() \mid = F$ in PAT, we verify whether the model satisfies the formula F.

Property 1: Deadlock Freedom

The system should not run into a deadlock state. We verify this property by means of a primitive in PAT.

$$\#assert\ System()\ deadlockfree;$$

Property 2: Data Availability

The property means that legal users should be able to obtain the required data. We define a Boolean variable $data\_suc$ to verify this property. When the subscriber gets the required data, we set the value of $data\_suc$ to $true$.

$$\#define\ \ Data\_Available\ \ data\_suc == true;$$
$$\#assert\ \ System()\ \ reaches\ \ Data\_Available;$$

Property 3: Data Leakage

Data leakage can cause a bad effect to the system. We use a Boolean variable dl to verify the property. If the intruder obtains the data, we set the value of dl to $true$.

$$\#define\ \ Data\_Leak\_Success\ \ dl == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ Data\_Leak\_Success;$$

Property 4: Device Faking

The property means that the intruder can pretend to be a legal device without being recognized. We adopt a Boolean variable df to verify the property. If the intruder fakes as a legal device successfully, we set the value of df to $true$.

$$\#define\ \ Device\_Fake\_Success\ \ df == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ Device\_Fake\_Success;$$

Property 5: User Privacy Leakage

User privacy leakage may bring great security risks to users. Hence, we check whether the intruder can obtain the sensitive information of the user using the following assertion.

$$\#define\ \ User\_Privacy\_Leak\ \ pl == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ User\_Privacy\_Leak;$$

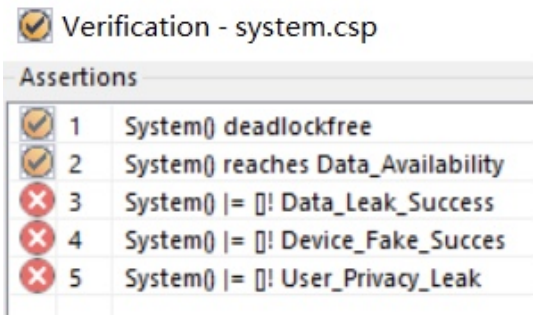


Fig. 5: Verification results of the original model

B. Verification Results

The verification results are shown in Fig. 5:

- $Property\ 1$ is valid. It represents that the model will never run into a deadlock state.
- $Property\ 2$ is valid. It shows that the data can be transmitted to the legal subscribers.
- $Property\ 3$ is invalid. It indicates that the intruder can obtain the data illegally.
- $Property\ 4$ is invalid. It means that the intruder can pretend to be a legal device to publish fake data.
- $Property\ 5$ is invalid. It indicates that the model cannot protect the user privacy once intruders appear.

C. Attack Analysis

According to the verification results, although AUPS adopts the access control and temporary keys, the system is still unreliable. Now we discuss the reasons for the insecure results. When the broker sends puk_b to the user, the intruder can intercept the message, and then replace puk_b with its public key puk_i. Since the user cannot detect that the key has been changed, the user sends sk_u and id_u encrypted with puk_i to the broker. Then the intruder can decrypt the message with prk_i to obtain sk_u and the user's sensitive information id_u, which leads to user privacy leakage. After obtaining sk_u, the intruder can get the data decryption key. Finally, the intruder can acquire the data, which results in data leakage. We give an example of the related attacks as follows:

$$A1.\ U \longrightarrow I : U.B.req_u$$
$$A2.\ I \longrightarrow B : U.B.req_u$$
$$A3.\ B \longrightarrow I : B.U.puk_b$$
$$A4.\ I \longrightarrow U : B.U.puk_i$$
$$A5.\ U \longrightarrow I : U.B.E(puk_i, sk_u.id_u)$$
$$A6.\ I \longrightarrow B : U.B.E(puk_b, sk_u.id_u)$$
$$A7.\ B \longrightarrow I : B.U.E(sk_u, sid_u.at)$$
$$A8.\ I \longrightarrow U : B.U.E(sk_u, sid_u.at)$$
$$A9.\ U \longrightarrow I : U.B.E(sk_u, at.T))$$
$$A10.\ I \longrightarrow B : U.B.E(sk_u, at.T))$$
$$A11.\ B \longrightarrow I : B.U.E(sk_u, kT)$$

where U, I and B mean user, intruder and broker respectively.

- $A1$: The user sends a request req_u to the broker.
- $A2$: The intruder intercepts the request.
- $A3$: The broker sends its public key puk_b to the user.

- $A4$: The intruder intercepts the message, and then replaces puk_b with its own public key puk_i.
- $A5$: The user sends its symmetric key sk_u and private information id_u encrypted with puk_i to the broker.
- $A6$: The intruder intercepts the message, and then decrypts the message to obtain sk_u and id_u using prk_i. At this point, user privacy leakage occurs.
- $A7$: The broker distributes the session identifier sid_u and attribute at encrypted with sk_u to the user.
- $A8$: The intruder acquires sid_u and at using sk_u.
- $A9$: The user requests to subscribe to service T.
- $A10$: The intruder eavesdrops on the message.
- $A11$: The broker distributes the data decryption key kT encrypted with sk_u to the user. The intruder intercepts the message and gets kT using sk_u. Then, the intruder can obtain the data using kT, which results in data leakage.

Similarly, the intruder can obtain the session identifier sid_d and symmetric key sk_d, and then fake as the device to publish data, which leads to device faking. We omit the details here.

D. Improved Model and Verification

In order to address the above issues, we improve the model by adding a digital certificate. Before sending the public key to other entities, the entity needs to send its public key to the Certification Authority (CA) to apply for a certificate. Fig. 6 depicts the flows of the digital certificate. First, the sender applies for a certificate. Second, CA generates a certificate based on the information of the sender, and then transmits the certificate to the receiver. Finally, the receiver verifies the validity of the certificate.

As the certificate is encrypted by CA's private key, the intruder cannot fake the certificate. It means that the intruder cannot replace the public keys of the honest entities with its own public key. Thus, the intruder can neither get the data nor violate user privacy. We modify the message definitions of the model. MSG_{key} is replaced by the following MSG_{key2}.

$$MSG_{key2} = \{msg_{key2}.a.b.E(k_1, k.inf) \mid a, b \in Entity, k_1, k \in Key, inf \in Inf\}$$

Then we formalize the improved processes of $Broker_1$, $User_1$, $Device_1$ and NOS_1 using the new message definitions. The improved model is given as follows.

$$System_1 =_{df} Broker_1 \| User_1 \| EF \| ProcessE \| KTM \| Device_1 \| NOS_1 \| CA \| Clock$$

$$System =_{df} System_1 [| INTRUDER\_PATH |] Intruder$$

The verification results are shown in Fig. 7. $Property\ 3-5$ are valid. It means that $Data\ Leakage$, $Device\ Faking$ and $User\ Privacy\ Leakage$ problems are solved now.

V. CONCLUSION AND FUTURE WORK

AUPS is an IoT system based on the publish/subscribe paradigm. In this paper, we formalized AUPS using the process algebra CSP. Feeding the model into PAT, we verified several functional and security properties of the model including deadlock freedom, data availability, data leakage, device faking and user privacy leakage. According to the verification results, data leakage, device faking and user privacy leakage may occur once intruders appear. Hence, we improved the model by using a digital certificate. Then we verified the improved model with PAT. The verification results show that the improved model can prevent intruders from invading the system. In the future, we will study more security properties of AUPS using formal methods and improve our model to handle more attacks.

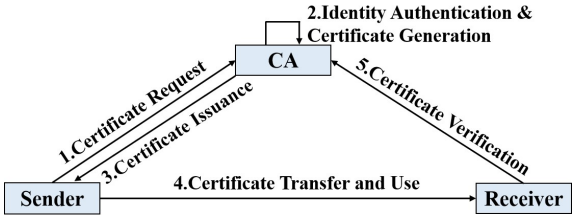


Fig. 6: Flows of digital certificate

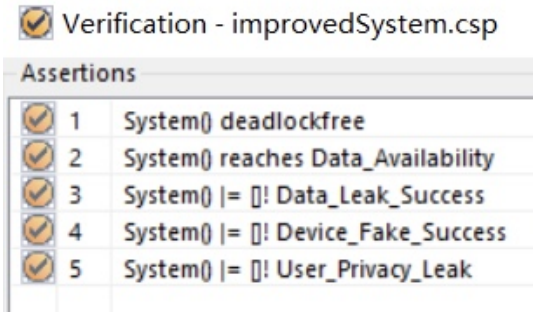


Fig. 7: Verification results of the original model

Acknowledgements. This work was partly supported by the National Key Research and Development Program of China (Grant No. 2018YFB2101300), the National Natural Science Foundation of China (Grant Nos. 61872145, 62032024), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing (East China Normal University).

REFERENCES

[1] Tewari A, Gupta B B. Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework. Future Gener. Comput. Syst. 2020, 108: 909-920.
[2] Khan F I, Hameed S. Understanding Security Requirements and Challenges in Internet of Things (IoTs): A Review. ArXiv abs/1808.10529 2019.
[3] Shi Y, Zhang Y, et al. Using machine learning to provide reliable differentiated services for IoT in SDN-like Publish/Subscribe middleware. Sensors, 2019, 19(6): 1449.
[4] Jung J, Choi Dong, et al. Distributed pub/sub model in CoAP-based Internet-of-Things networks. *Proc.* of the International Conference on Information Networking (ICOIN), 2018: 657-662.
[5] Shang W, Gawande A, et al. Publish-Subscribe Communication in Building Management Systems over Named Data Networking. *Proc.* of the 28th International Conference on Computer Communication and Networks, 2019: 1-10.
[6] Rizzardi A, Sicari S, et al. AUPS: An Open Source AUthenticated Publish/Subscribe system for the Internet of Things. Inf. Syst. 2016, 62: 29-41.
[7] Hu V C, Kuhn D R, et al. Attribute-based access control. Computer, 2015, 48(2): 85-88.
[8] Hoare C A R. Communicating sequential processes. Communications of the ACM, 1978, 21(8): 666-677.
[9] PAT, PAT: Process Analysis Toolkit. 2019. http://pat.comp.nus.edu.sg.

Formal Verification of the Lim-Jeong-Park-Lee Autonomous Vehicle Control Protocol using the OTS/CafeOBJ Method

Tatsuya Igarashi Masaki Nakamura Kazutoshi Sakakibara

Toyama Prefectural University, Toyama, Japan

Abstract- **The Lim-Jeong-Park-Lee protocol (LJPL protocol) has been proposed as an efficient distributed mutual exclusion algorithm for intersection traffic control. The LJPL protocol has been specified and verified formally using the Maude model checker. Because of the limitation of computation, the existing model checking approach restricts the number of vehicles participating the protocol. In this paper, we model the LJPL protocol as an observational transition system, describe its specification in CafeOBJ, the algebraic specification language, and verify its safety property using the proof score method, where mutual exclusiveness can be proved for an arbitrary number of vehicles** [*].

Keywords-component; autonomous vehicles; the Lim-Jeong-Park-Lee protocol; algebraic specification; observational transition system; proof score method

I. Introduction

In [1], an efficient distributed mutual exclusion algorithm for intersection traffic control, called the Lim-Jeong-Park-Lee protocol (LJPL protocol), has been proposed, where each lane of the intersection has a queue of vehicles (Figure 1). All vehicles in a queue can enter the intersection if the top vehicle of the queue arrived first among the waiting vehicles in the other conflict lanes. Since the vehicles except the top one do not need extra permissions to enter the intersection, the protocol has been shown to be effective.

In [2], the LJPL protocol has been formally specified and some properties are verified by Maude tool[†]. In Maude, a state transition system is specified as a rewrite specification. A desired property is verified by fully automated model checking. In principle, model checking restricts the state space finite. Thus, by the Maude model, only finite combinations of initial states can be treated. In [2], it is mentioned that an initial state with five vehicles has been proved to be safe and the authors had encountered the state explosion problem for the case of more than a dozen vehicles.

In our study, we model the LJPL protocol as an observational transition system (OTS) [3, 4, 5], where a state of the system is not represented explicitly but can be identified through a given set of observation functions. A state transition is also defined through observations. By such an approach, we may obtain more abstract system specifications independent from the structure of states. Especially our model does not fix the number of vehicles participating the protocol. An OTS can be specified in CafeOBJ language[‡], which supports not only specification description based on equational specifications but also specification execution based on term rewriting theory. Roughly speaking, when we add a new equation $t_0 = t_1$ to a given specification SP, reduce a term t_2 by the CafeOBJ processor and obtain t_3 as a reduced term, then it guarantees that the implication $t_0 = t_1 \Rightarrow t_2 = t_3$ holds for all models of the specification. By combining specification executions, we may construct complicated proofs, such as case splitting and inductions. To make a complete proof through interaction with CafeOBJ processor is called the proof score method, or the OTS/CafeOBJ method. For the OTS/CafeOBJ specification of the LJPL protocol, we verify the safety property such that vehicles of different conflict lanes cannot enter at same time by the proof score method.

II. Lim-Jeong-Park-Lee protocol

We give a brief introduction of the LJPL protocol in this section. See [1, 2] for more detail.

The intersection of the LJPK protocol is a crossroad represented in Figure 1. The lanes are labeled by lane0 , . . . , lane7. Each of four directions has two lane: the straight or right turn lanes (even numbered) and the left turn lanes (odd numbered). When some vehicle is crossing the intersection, some vehicle can enter the intersection and some are not. A lane l conflicts with a lane l' if a vehicle in l may collide with a vehicle in l' when they enter the intersection at same time. For example, lane0 conflicts with lane2, lane5, lane6 and lane7.

[*]DOI reference number: 10.18293/SEKE2022-028.

This work was supported by JSPS KAKENHI Number JP19K11842.

[†]http://maude.cs.uiuc.edu

[‡]https://cafeobj.org/intro/ja/

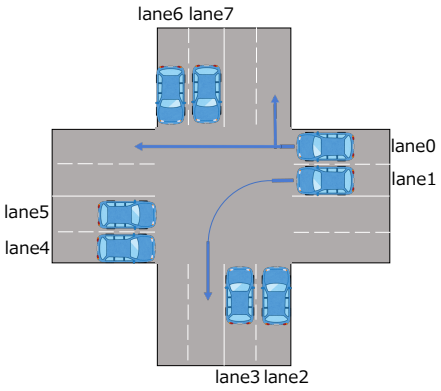


Figure 1. The intersection of the LJPL protocol

In the LJPL protocol, a vehicle passes the intersection through the following states: *running*, *approaching*, *stopped*, *crossing* and *crossed*. In the running state, the vehicle is running before the intersection. From the *running* state to the *approaching* state, the vehicle approaches the queue of a lane. From *approaching* to *stopped*, the vehicle is added to the queue and the arrival time of the top vehicle of the queue is set to the vehicle. From *stopped* to *crossing*, the vehicle enters the intersection if the time of the vehicle is less than the time of the top vehicle of each conflict lane. From *crossing* to *crossed*, the vehicle leaves the intersection.

III. An OTS/CafeOBJ specification of the LJPL protocol

In this section, we give an OTS/CafeOBJ specification of the LJPL protocol. We assume the reader is familiar with observational transition systems and CafeOBJ algebraic specification language, and introduce the notions and notations briefly through the specification of the LJPL protocol. See [3, 4, 5] for full syntax and semantics of OTS/-CafeOBJ specifications.

A Data modules

An OTS/CafeOBJ specification consists of data modules and a system module. We first give a data module `VID` for vehicles.

```
mod* VID{
  [Vid < Vid+]
  op dummy : -> Vid+
  op _=_ : Vid+ Vid+ -> Bool {comm}
  eq (I:Vid = dummy) = false .
  eq (V:Vid+ = V) = true . }
```

The module declaration with `mod*` denotes the loose denotation, where the module denotes all models (algebras) which satisfies all equations in the modules. The name of the module is `VID`. Two sorts `Vid` and `Vid+` are declared with a relationship `Vid` < `Vid+`. Each sort denotes a (carrier) set in a model. Hereafter we deal with a sort as a set if no confusion occurs. Sort `Vid` is interpreted as a subset of `Vid+`. We intend to use `Vid` as a set of (identifiers of) vehicles and `Vid+` as a set of vehicles including a dummy one. There are two operator declarations with `op`. The name of the first operator is `dummy`, which takes the empty arity and returns `Vid+`. The empty arity operator denotes a constant of the returned sort. The commutative operator `_=_` takes two arguments of `Vid+` and returns a boolean value. There are two equations which all models satisfy. A term $X{:}S$ is a variable of Sort S, which denotes an arbitrary element of the sort. The first equation declare that all elements of `Vid` are not equivalent to the dummy vehicle. By the second equation, each vehicle is equivalent to itself.

The following is a data module for (identifiers of) lanes. For all $i, j \in \{0, 1, \ldots, 7, 999\}$, we give the values of `lane`i = `lane`j and `lane`i < `lane`j by equations. The dots part (`...`) are omitted.

```
mod! LID{
  [Lid]
  ops lane0 lane1 lane2 lane3 lane4 lane5 lane6 lane7
      lane999 : -> Lid
  op _=_ : Lid Lid -> Bool {comm}
  op _<_ : Lid Lid -> Bool
  eq (L:Lid = L) = true .
  eq (lane0 = lane1) = false .  ...
  eq (L:Lid < L) = false .
  eq (lane0 < lane1) = true . ...
  eq (lane5 < lane3) = false . ... }
```

The module declaration with `mod!` denotes the tight denotation, where the module denotes only the initial model, where each element of the model has a corresponding term constructed from operators in the module (no duplication), and an equation is deducible from the equations of the module, whenever the both hand sides of the equation are interpreted into a same element (no confusion). In the model of `LID`, Sort `Lid` has exactly nine elements of `lane0`~`lane999`. The constants `lane0`~`lane7` stand for lanes of the intersection. The constant `lane999` stands for a special lane where all vehicles belong before coming the intersection. We define the order of lanes as `lane` i is smaller than `lane` j iff $i < j$.

We specify a tight data module `VSTAT` of the labels of vehicles' states, where Sort `Vstat` and constants `running`, `approaching`, `stopped`, `crossing` and `crossed` of Sort `Vstat` are declared with an equivalent predicate `_=_` similarly. We also specify a data module `TIMEVAL` with the built-in sort `Rat` of rational numbers, Sort `Rat+` of rational numbers with the infinity `oo` and predicates `_<_`, `_<=_`, `_=_` on `Rat+`. We omit the details of `VSTAT` and `TIMEVAL`.

Since the LJPL protocol manages a queue of vehicles, we specify a data module `QUEUE` of queues as follows:

```
mod! QUEUE{
  pr(VID)  [Queue]
  op empty : -> Queue
```

```
  op _,_ : Vid Queue -> Queue
  op put : Queue Vid -> Queue
  op remove : Queue -> Queue
  op top : Queue -> Vid+ ... }
```

Module QUEUE imports Module VID with the protect mode, where a model (carrier sets) of the importing module includes a model of the imported module as it is. The first two operators empty and _,_ are constructors of queues. The set Queue of queues is defined inductively. Constant empty denotes the empty queue. Term e,*queue* is a queue whose top element is e and the tail is *queue* if e is of Vid and *queue* is of Queue. For example, Term e_0,e_1,e_2,empty is a term of Queue. Operator put, remove and top are standard operations of queues. Term put(queue,e) stands for the result queue by adding an element e to a *queue* as the last element. Term remove(queue) is the result queue by deleting the top of *queue*. Term top(queue) is the top element of *queue*. For example, Operator put is defined by the following equations in QUEUE:

```
  eq put(empty,I:Vid) = I,empty .
  eq put((J:Vid,Q:Queue),I:Vid) = J,put(Q,I) .
```

For example, Term put((a,b,empty),c) is equivalent to Term a,b,c,empty since put((a,b,empty),c) = a,put((b,empty),c) = a,b,put(empty,c) = a,b,c,empty. Similarly, the other operators are defined inductively.

B *The system module : observers*

We give a system module of the LJPL protocol. First, we give observers and a definition of an initial state of our system module.

```
mod* OTS{
  pr(LID + VSTAT + QUEUE + TIMEVAL)
  *[Sys]*
  bop lid : Sys Vid+ -> Lid
  bop vstat : Sys Vid+ -> VStat
  bop t : Sys Vid+ -> Rat
  bop lt : Sys Vid+ -> Rat+
  bop q : Sys Lid -> Queue
  bop now : Sys -> Rat
```

Our system module, OTS, imports all data modules defined above with the protecting mode. Sort Sys is declared as a hidden sort, which denotes the state space of the system. An operator with the hidden sort in its arity is called a behavioral operator. A behavioral operator is divided into two categories: it is called an observer if the returned sort is not hidden and a transition if it is hidden. Six observers are declared in OTS: lid(s,i) is the lane ID of a vehicle i at a state s. vstat(s,i), t(s,i) and lt(s,i) are the state, the arrival time, and the arrival time of the top vehicle in the queue of the lane of a vehicle i at a state s respectively. q(s,l) is the queue of a lane l. now(s) is the elapsed time at a state s.

The following specifies an initial state.

```
  op init : -> Sys
  eq lid(init,I:Vid) = lane999 .
  eq vstat(init,I:Vid) = running .
  eq t(init,I:Vid) = oo .
  eq lt(init,I:Vid) = oo .
  eq q(init,L:Lid) = empty .
  eq now(init) = 0 .
```

Constant init is an element of Sys, which we call the initial state. The initial state is not defined explicitly but is defined through observers. The first equation specifies the initial lane of all vehicles is lane999. The state is defined as running. The arrival times are defined as the infinity oo, which means that they have the lowest precedence to enter the intersection.

For the dummy vehicle, its lane, state, arrival times are defined as lane999, stoped and oo for all states respectively. The queue of lane999 is defined as empty for all states. We omit the equations for the dummy vehicle.

C *The system module : transitions*

State transitions are declared as follows:

```
  bop set : Sys Vid Lid -> Sys
  bop approach : Sys Vid -> Sys
  bop check : Sys Vid -> Sys
  bop enter : Sys Vid -> Sys
  bop leave : Sys Vid -> Sys
  bop tick : Sys Rat -> Sys
```

Term set(s,i,l) is the result state after applying the transition set for a vehicle i and a lane l at the state s. Similarly, other transitions are declared as operators which take a current state and return the result state with some parameters. Transition tick(s,x) is a special transition which advances elapsed time by x.

For a transition τ, the effective condition c-τ is a condition under which the transition τ can be applied. The following is a definition of the effective condition c-set of the transition set [§].

```
  op c-set : Sys Vid -> Bool
  eq c-set(S:Sys,I) =
     (vstat(S,I) = running && lid(S,I) = lane999) .
  ceq set(S,I,L) = S if not c-set(S,I) .
```

The operator c-set is declared and is defined by the first equation such that set is effective for a vehicle I if I's state is running and lane is lane999. The last equation is a conditional equation, where the body equation holds when the condition part is true. The condition part of the last equation is not c-set(S,I), that is, set is not effective for I. Then, the body equation says that the result of applying set does not change a state. The application of the transition is considered to be ignored when it is not effective.

The following is the set of all equations defining set when it is effective.

[§] Hereafter we use D, I, J, S and L (L') as variables of Rat, Vid, Vid+, Sys and Lid respectively.

```
  ceq lid(set(S,I,L),J) =
      (if I = J then L else lid(S,J) fi)
      if c-set(S,I) .
  ceq vstat(set(S,I,L),J) = vstat(S,J) if c-set(S,I) .
  ceq t(set(S,I,L),J) = t(S,J) if c-set(S,I) .
  ceq lt(set(S,I,L),J) = lt(S,J) if c-set(S,I) .
  ceq q(set(S,I,L),L') = q(S,L') if c-set(S,I) .
  ceq now(set(S,I,L)) = now(S) if c-set(S,I) .
```

The first equation specifies that the lane ID of a vehicle `J` after `set(S,I,L)` is defined as `L` if `I = J` when `set` is effective, and it is unchanged if `I ≠ J`. Only lane ID is changed and the other observed values are unchanged as defined by the following five equations. By Transition `set`, a vehicle can be assigned to any lane.

To define a system behavior completely, for all combinations of an observer o and a transition τ, we need to define the value observed by o of the result state after applying τ to a state s, denoted by $o(\tau(s))$. Since there are lots of equations in our system module, we show subset of them in this paper.

Transition `approach` is defined as follows:

```
eq c-approach(S,I) =
  (vstat(S,I) = running && not(lid(S,I) = lane999)) .
ceq vstat(approach(S,I),J) = (if I = J then approaching
    else vstat(S,J) fi) if c-approach(S,I) .
ceq t(approach(S,I),J) = (if I = J then now(S)
    else t(S,J) fi) if c-approach(S,I) .
ceq q(approach(S,I),L) = (if L = lid(S,I) then
    put(q(S,L),I) else q(S,L) fi) if c-approach(S,I) .
```

Transition `approach` is effective if the state is in `running` and the lane is not `lane999`, that is, immediately after `set`. By `approach(S,I)`, the state of `I` becomes `approaching`. The arrival time is set to the current time `now(S)` and `I` is added to the queue of the belonging lane.

Transition `check` is defined as follows:

```
eq c-check(S,I) = (vstat(S,I) = approaching &&
      top(q(S,lid(S,I))) = I
   || vstat(S,getpre(q(S,lid(S,I)),I)) = stopped
   || vstat(S,getpre(q(S,lid(S,I)),I)) = crossing)) .
ceq vstat(check(S,I),J) = (if I = J then stopped
    else vstat(S,J) fi) if c-check(S,I) .
ceq lt(check(S,I),J) = (if
    I = J && (top(q(S,lid(S,I))) = I
    || vstat(S,getpre(q(S,lid(S,I)),I)) = crossing)
    then t(S,I) else if
    I = J && vstat(S,getpre(q(S,lid(S,I)),I)) = stopped
    then lt(S,getpre(q(S,lid(S,I)),I)) else lt(S,J)
    fi fi) if c-check(S,I) .
```

Transition `check` is effective when the state of the vehicle is `approaching` and either it is top of the queue or the previous vehicle's state is `stopped` or `crossing`, where `getpre(`q`,`i`)` returns the previous vehicle of i in a queue q. The state of the vehicle becomes `stopped`. The last equation specifies that the arrival time of the previous vehicle in the queue is set to the vehicle as `lt`.

Transition `enter` is defined as follows:

```
eq c-enter(S,I) = (vstat(S,I) = stopped
                   && top(q(S,lid(S,I))) = I && ...)
ceq vstat(enter(S,I),J) = (if lid(S,I) = lid(S,J) &&
    vstat(S,J) = stopped then crossing else
    vstat(S,J) fi) if c-enter(S,I) .
```

Transition `enter` is effective when the vehicle's state is `stopped`, it is top of the queue and the arrival time `lt` is smaller than that of the top vehicle of each conflict lane. We omit a part of the right-hand side of the first equation of `c-enter`. The states of all vehicles in the same queue (`stopped`) become `crossing`, that is, they enter the intersection at once.

Transition `leave` is defined as follows:

```
eq c-leave(S,I) =
  (vstat(S,I) = crossing && top(q(S,lid(S,I))) = I) .
ceq vstat(leave(S,I),J) = (if I = J then crossed
    else vstat(S,J) fi) if c-leave(S,I) .
ceq q(leave(S,I),L) = (if L = lid(S,I) then
    remove(q(S,L)) else q(S,L) fi) if c-leave(S,I) .
```

Transition `leave` is effective when the vehicle's state is `crossing` and it is top of the queue. The vehicle's state becomes `crossed` and it is removed from the queue.

Finally, Transition `tick` is defined as follows:

```
eq now(tick(S,D)) = now(S) + D .
```

Transition `tick(S,D)` is always effective and it increase the current time by `D`.

D Specification execution

The CafeOBJ reduction command reduces a term to a term equivalent to the input term based on term rewriting theory. The following is an example of reduction.

```
open OTS . ...
eq s1 = set(set(set(set(
        init,a,lane0),b,lane3),b2,lane3),c,lane4) .
eq s2 = approach(approach(approach(s1,a),b),c) .
eq s3 = approach(tick(s2,1),b2) .
eq s4 = check(check(check(check(s3,a),b),c),b2).
eq s5 = enter(enter(enter(s4,c),b),a) .
red vstat(s5,a) . red vstat(s5,b) .
red vstat(s5,b2) . red vstat(s5,c) .
close .
```

State `s1` is equivalent to a term obtained by applying four `set` transitions with vehicles `a`, `b`, `b2`, `c` with lanes `lane0`, `lane3`, `lane3`, `lane5` respectively. We apply Transition `approach` to vehicles `a`, `b`, `c`, and advance time by one time unit and apply `approach` to `b2` (State `s3`). Then, we apply `check` to all vehicles and apply `enter` to `a`, `b`, `c`. State `s5` is the result state. Note that lane `lane0` does not conflict with lane `lane3` and `lane4` but lane `lane3` conflict with `lane4`.

By the last four reduction commands, we check states of all vehicles. CafeOBJ returns `crossing` for `a` in `lane0`, `crossing` for `b` and `b2` in `lane3`, and `stopped` for `c` in `lane4`. Vehicles `a` and `b`, `b2` are `crossing` since they do not conflict with each other. Vehicle `c` failed to enter (from `stopped` to `crossing`) since `b` in a conflict lane has already entered. Although `b2`'s arrival time is later than `c`'s arrival time, `b2` entered since it belongs to the same queue with `b`.

IV. Formal verification of the LJPL protocol

In this section we verify the safety property of the LJPL protocol, that is, no two vehicles enter the intersection if they belong to conflict lanes, by using the proof score method. First we formalize a safe state by operators and equations.

```
mod INV{ pr(OTS) ...
eq concur(L,L') = ((L = L') ||
  (L = lane0 && (L' = lane1 || L' = lane3 || L' = lane4))
  || ... .
eq inv1(S,I,J) = (not(I = J)
   && vstat(S,I) = crossing && vstat(S,J) = crossing)
   implies concur(lid(S,I),lid(S,J)) . }
```

The first equation specifies a predicate `concur` such that `concur(L,L')` is true if lanes `L` does not conflict with `L'`. Then, the invariant property `inv1` is defined by the last equation. The invariant property `inv1(S,I,J)` is true if vehicles `I` and `J` do not belong to conflict lanes whenever `I` and `J` are different and their states are `crossing`. The invariant property is a state predicate. If `inv1(`s`,`i`,`j`)` is true for all states s reachable from the initial state and vehicles i and j, the LJPL protocol is safe. In OTS/CafeOBJ specifications, reachable states are represented by terms like $\tau_n(\cdots(\tau_1(\tau_0(\texttt{init}))))$, which stands for the result state after applying transitions $\tau_0, \tau_1, \ldots, \tau_n$ to the initial state in this order. Since reachable terms are infinite, we prove this claim by induction on the structure of reachable states. The base step is proved for the initial state `init` and the induction step is proved for $s' = \tau(s)$ for each transition τ with the assumption of `inv1(`s`,`i`,`j`)` as the induction hypothesis.

Base step The following is a fragment of a proof score, called a proof passage, for the base step.

```
open INV .
ops i j : -> Vid .
red inv1(init,i,j) .
close .
```

Constants `i` and `j` are declared as arbitrary vehicles. The reduction command `red` takes a term and returns a term reduced by using declared equations. CafeOBJ processor returns `true` as the result of the above reduction, that guarantees that the base step is proved successfully.

Induction step The following is a module for proving induction steps.

```
mod ISTEP{ pr(INV) ...
  ops s s' : -> Sys
  eq istep1(I,J) = inv1(s,I,J) implies inv1(s',I,J) . }
```

Constants `s` and `s'` are declared as arbitrary states. For each induction step of a transition τ, we declare an equation `s'` $= \tau($`s`$)$. Thus, in induction steps, we prove the implication `inv1(s,`i`,`j`)` $\Rightarrow$ `inv1(s',`i`,`j`)` for each vehicles i and j. Predicate `istep1` is declared for proving the implication.

The following is a proof passage for Transition `set` in the case that the effective condition is false.

```
open ISTEP .
ops i j k : -> Vid . op l : -> Lid .
eq c-set(s,k) = false .
eq s' = set(s,k,l) .
red istep1(i,j) .
close .
```

The above reduction returns `true`. Thus, if `set` is not effective, the induction step for `set` is proved. The following is the case that it is effective.

```
open ISTEP .
ops i j k : -> Vid .
op l : -> Lid .
eq vstat(s,k) = running .
eq lid(s,k) = lane999 .
eq s' = set(s,k,l) .
red istep1(i,j) .
close .
```

Note that we declare two equations instead of `c-set(s,k) = true`. They are same meaning from the definition of `c-set` in the system module. Unfortunately, the above reduction does not return `true`. The result of the reduction is a term like `(if (k = i) then l else lid(s,i) fi) = (if (k = j) then ...) ...`. This result means that CafeOBJ cannot prove the input property to be true or false. In such a case, we revise the proof passage such that CafeOBJ can prove it. Such a procedure is called an interactive theorem proving.

In this case, the result term includes `k = i`. If it is true or false, CafeOBJ may proceed reduction more. Thus, we apply a case splitting about `k = i`. We make two copies of the above failed proof passage, add equations `k = i` and `(k = i) = false` for each copy. Since `k = i` $\vee$ `(k = i) = false` = *true*, if the both copies return true then the original proof passage is true. If results are not true or false, we apply case splitting until it is reduced into true or false.

Lemma discovery If a proof passage returns false, there are two possibilities, the invariant property is not true or the considered state is unreachable from the initial state.

Consider the following proof passage which returns false.

```
open ISTEP .
ops i j k : -> Vid .
eq vstat(s,k) = stopped . eq top(q(s,lid(s,k))) = k .
eq lid(s,k) = lane0 .     eq top(q(s,lane0)) = k .
eq vstat(s,top(q(s,lane2))) = stopped . ...
eq s' = enter(s,k) .
eq i = k . eq (j = k) = false . eq lid(s,j) = lane2 .
eq (lid(s,j)= lane0) = false . eq vstat(s,j) = crossing .
red istep1(i,j) .
close .
```

In this case, the vehicle `i = k` is the top of Lane `lane0` and waits for `enter`. Although the vehicle `j` is `crossing` in `lane2`, the top of `lane2` is `stopped`. In the LJPL protocol, a vehicle in a queue should not be the state of `crossing` if

the top vehicle of its lane is in the state of `stopped`. Thus, this case of the proof passage is considered to be unreachable state from the initial state.

To solve this proof passage, we introduce a lemma extracted from the unreachable state. The following is a lemma we introduce.

```
eq pred1(S,empty) = true .
eq pred1(S,(I,Q)) = (if vstat(S,I) = crossing
                     then false else pred1(S,Q) fi) .
eq pred2(S,empty) = true .
eq pred2(S,(I,Q)) = (if vstat(S,I) = stopped
                     then pred1(S,Q) else pred2(S,Q) fi) .
eq inv2(S,I) = pred2(S,q(S,lid(S,I))) .
```

Predicate `inv2` is the lemma we introduce and Predicates `pred1` and `pred2` are auxiliary predicates for defining the lemma. Predicate `pred1(S,Q)` is true if the queue `Q` does not have `crossing` vehicles. Predicate `pred2(S,Q)` is true if no `crossing` vehicles exist after any `stopped` vehicle. The invariant `inv2(S,I)` is defined by `pred2` with State `S` and the queue of the lane of Vehicle `I`. We add the invariant to the proof passage as the premise of the target implication as follows:

```
open ISTEP . ...
eq (lid(s,j)= lane0) = false . eq vstat(s,j) = crossing .
red inv2(s,j) implies istep1(i,j) .
close .
```

Then, the reduction does not return false. We proceed case splitting and lemma discovery repeatedly and all proof passages (cases) for `inv1` become true after introducing more two lemmata `inv3` and `inv4`.

```
eq inv3(S,I) =
  (vstat(S,I) = approaching || vstat(S,I) = stopped ||
   vstat(S,I) = crossing) implies (I in q(S,lid(S,I))) .
eq inv4(S,I) = vstat(S,I) = crossing implies
   not pred1(S,q(S,lid(S,I))) .
```

Verification of lemmata In the previous section we showed the main invariant property `inv1` holds under the assumption of three lemmata. In order to complete a proof we need to prove those lemmata. They can be proved by the induction on reachable states similarly. Although we do not need more lemmata about the induction on reachable states, we needed to introduce another kind of lemmata, for example, `pred2(set(s,k,L),q)` = `pred2(s,q)`, which can be proved by the induction on the data structure of queues `q`.

Finally, we obtain a complete proof score for `inv1` with 609 proof passages which all return true, where three lemmata about reachable states and 17 lemmata about queues are introduced. Since the data module `VID` of vehicles denotes the loose denotation, the system specification `OTS` denotes all systems following the LJPL protocol with arbitrary number of vehicles. Our verification result guarantees that the LJPL protocol is safe for any vehicles.

V. Conclusion

We described an OTS model of the LJPL protocol in CafeOBJ language and verified a safety property by the proof score method. The main contribution of our study is to give a formal verification of the safety property of the LJPL protocol for arbitrary number of vehicles.

Through the experience of formal verification of the LJPL protocol, we faced lemmata about queues as well as lemmata `inv1`∼`inv4` about reachable states. Although to find an appropriate lemma about reachable states we may need an insight into a target system, the lemmata about queues seem to have some pattern. To investigate a way to construct a semi-automated support tool for the proof score method for such data types is one of our future work.

In [2], not only the safety property we deal with in this study but other important properties of intersection control protocols have also been verified, e.g. the deadlock-freedom and the starvation-freedom properties. To specify and verify such properties in our OTS/CafeOBJ specification is another one of our future work.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP19K11842.

References

[1] J. Lim, Y. Jeong, D. Park, and H. Lee, An efficient distributed mutual exclusion algorithm for intersection traffic control, The Journal of Supercomputing, vol.74, pp.1090-1107, 2018.

[2] Moe Nandi Aung, Yati Phyo, Kazuhiro Ogata, Formal Specification and Model Checking of the Lim-Jeong-Park-Lee Autonomous Vehicle Intersection Control Protocol, SEKE 2019, pp.159-208, 2019.

[3] K. Ogata, and K. Futatsugi, Proof scores in the OTS/CafeOBJ method, FMOODS 2003, LNCS 2884, pp.170-184. Springer, 2003.

[4] K. Ogata and K. Futatsugi, Modeling and verification of real-time systems based on equations, Science of computer programming, 66(2), pp.162-180, 2007.

[5] Masaki Nakamura, Shuki Higashi, Kazutoshi Sakakibara, Kazuhiro Ogata, Specification and verification of multitask real-time systems using the OTS/-CafeOBJ method, IEICE Transactions on Information and Systems, Vol.E105-A, No.5, pp.-, 2022. (accepted)

Improving the Early Rumor Detection Performance of the Deep Learning Models By CGAN

Fangmin Dong[2], Yumin Zhu[2], Shuzhen Wan[2], Yichun Xu[1,2*]
[1]Hubei Province Engineering Technology Research Center for Construction Quality Testing Equipments
[2]College of Computer and Information Technology
China Three Gorges University, Yichang 443002, Hubei, China
*Corresponding author email: xuyichun@ctgu.edu.cn

***Abstract*—Deep learning models are recently applied to detect rumors on social media based on the information in the posts. However, at the early stage of rumor propagation, due to the lack of responses, the performance of these models often degrades. In this paper, we propose a method based on the conditional generative adversarial network, which can generate the responses like data and help the deep learning models in early detection. On two large-scale Sina Weibo datasets, the proposed method is applied on the existing convolution neural network model, the recurrent neural network model, and the recursive neural network model. The results show that the proposed method can significantly improve the performance of the models in the case of zero response, and has performance superiority in a certain early period.**

Keywords-Rumor Detection, Deep Learning, Conditional Generative Adversarial Network, Sina Weibo

I. INTRODUCTION1

The nowadays social media provided a place for the people to spread his statements rapidly on the internet; however, the rumors have also proliferated there. The proliferation of rumors has caused significant damage to individuals and society [1]. The social media operators and the government have established the platforms to deal with the rumors, such as WeiboPiyao, but these platforms rely on manual inspection, so they are inefficient. In order to improve the efficiency, machine learning technology has been applied to rumor detection, such as support vector machine, decision tree, and logistic regression [2]. These models work on the data features of the events, which are extracted from the texts and images within the posts, the user profiles, and the propagation structure. However, the data features are also manually extracted and the feature engineering is painstakingly detailed, biased, and labor-intensive. Recently, the deep learning techniques dispense with the complex manual feature extraction and perform the rumor detection by the neural networks, such as conventional neural networks (CNN) [3], recurrent neural networks (RNN) [4], tree recurrent neural networks (RvNN) [5, 6]. The reported results show that the deep learning models generally have better performance compared to traditional machine learning models.

This work was supported by the National Science Foundation of China (U1703261).
DOI reference number: 10.18293/SEKE2022-030

Early detection is very important because it can reduce the damages by the rumor propagation. It should be ideal to make the detection when a user has just posted a source post and no any other user follows him (we later call this case as detection with zero response). On the other hand, since the aforementioned learning models are trained on the events with hundreds of responses, when they make the predictions with less or no response, their performance will certainly decline, because most important patterns may not appear at that time. In comparison, the reported results showed that the CNN or RNN models [3, 4, 7] had better early performance than the traditional methods, and the recent RvNN models [5, 6] gave the improved early detection results above the CNN and RNN models. However, the early detection performances of these models are still a little weak.

In this paper, we propose a method that can improve the early detection performance of some deep learning models. We believe that some conditional relationships exist between the source (first) post and the responses, so that we train a conditional generative adversarial network (CGAN) to generate the response data from source post. In early detection, when there is only a source post or with few responses, the generated data is added to assist the detection. The idea behind is that we adjust the test data and try to make it accord with the distribution of the training data.

On two public Sina Weibo datasets, we applied the proposed methods on the existing CNN model [3], LSTM model [4] and RvNN model [5] respectively. The results show that the improved CNN and RNN models are nearly exceed the original RvNN model in the early detection, and moreover, the RvNN model can also be improved.

II. RELATED WORKS

A. Deep Learning Models for Rumor Detection

In this section, we review the related deep learning models for rumor detection.

CNNs. Convolutional neural network (CNN) is a multi-layer network that uses local connectivity and shared weights to reduce the network complexity and has been applied very successfully in image classification [8]. CNN extracts features by convolutional and pooling computations, and they are suitable for the structured data, and were also used in text classification [9]. Liu et al. used word vector to represents the posts and responses and then applied CNN to perform rumor

detection [10]. Yu et al. divided the posts into groups by time and used doc vector to represent them, finally used CNN to extract high level information and achieved rumor detection [3].

RNNs. Recurrent neural network (RNN) takes sequence data as input, and performs recursive computation in the sequence direction, thus it can capture the historical information of the sequence. Ma et al. first applied RNN to rumor detection and discussed the performance of different RNN types, such as gate recurrent unit (GRU) and long short-term memory (LSTM) [4]. Chen et al. used LSTM network with soft attention mechanism [7]. Xu et al. used user information to perform data preprocessing to improve the performance of RNN [11].

Some other works combined CNN and RNN on the rumor detection. Nguyen et al. used CNN and RNN sequentially to score the reliability of a single tweet and then make the rumor detection by a time series model [12]. Liu et al. made the rumor detection only on the user data, that the data were processed and synthesized by both CNN and RNN [13].

RvNN. Unlike the RNN which performs chain recursion on sequence data, a recursive neural network (RvNN) which makes recursion on a tree structure was proposed by Ma et al. [5, 6]. They found that the propagation of information forms a tree, and the rumor tree and the non-rumor tree have different node relationships-the support and opposition structures are different. So they proposed a tree-structured RvNN to extract these relationships. Recently, the propagation of information were handled with the graph convolutional networks [14-16], where the tree structure were also used.

B. Generative Adversarial Network for Rumor Detection

Generative adversarial network (GAN) was proposed by Goodfellow et al. [17], consisting of a generator and a discriminator, where the generator generates fake data and the discriminator tries to distinguish between the real and fake data. During the training process, the generator and the discriminator compete with each other and the final generated data has the similar distribution as the real data. Currently GAN has been widely used in computer vision, natural language processing, and artificial intelligence [18]. When some conditional parameters are added to the input, we get a conditional generative adversarial network (CGAN), that the data can be generated according to the conditions [19, 20].

GANs were applied in some classification problems, where they were used to generate more data to help the training of the classifiers [21, 22]. But in this paper, we apply the GAN in a different way that the generated data is used in the prediction phase.

The applications of GAN have also been found in rumor detection field. Wang et al. used a GAN to remove the specificity in different types of events, thus their model can extract event-independent features and get good performance [23]. Considering that some rumor mongers tried to mislead the public with pseudo-responses, Ma et al. built a generator to insert pseudo-responses and thus strengthen the discriminator [24]. Song et al build adversary generator by encoder-decoder framework to produce a response for malicious attack [14]. However, the objectives of these works are not related to the early detection.

III. PROBLEM STATEMENT

In social media, most rumors are spreading only with textual information, and the techniques based on texts are the focus of rumor detection field. Other techniques based on multimedia information also need the texts. So in this paper we mainly conducted the rumor detection by the text of the posts. Some relevant concepts are listed below:

Post: The text message posted by a user in social media with a limited number of words. The first post about an event is called the source post;

Response: A comment post made by a user after he read a post. A response can be a comment to the source post, or to a response;

After a user posts a source post P_0 on social media, the post is then read and responded by several other users. Suppose there are N responses to the source post, which are P_1, P_2 ..., and P_N in time order. The sequence E=P_0, P_1, P_2 ..., P_N is called an event.

The machine learning algorithm uses the event data to train a model M, and then uses M to predict a new event being a rumor or not. The early rumor detection takes place at the early stage of propagation, that the number of responses N is very small, even zero. Original models trained with full-life event data will suffer performance degrade in early detection due to the lack of responses. In this paper, we integrate CGAN into the original model to improve its early detection performance.

IV. EARLY RUMOR DETECTION METHOD

A. Overall Framework

The proposed method includes three main modules, which are illustrated in Fig. 1 with different colors.

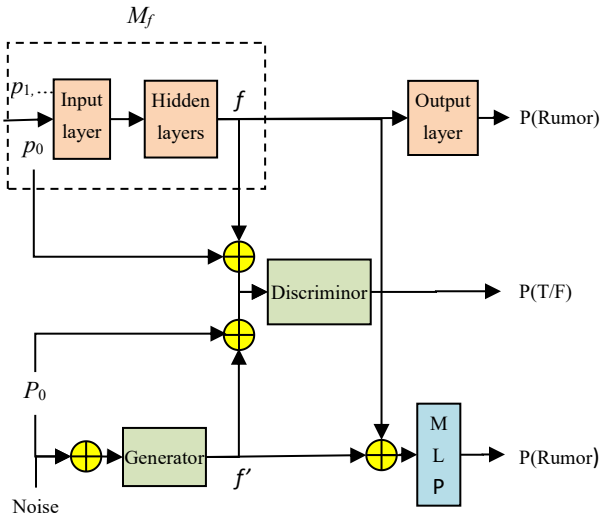


Figure 1. The framework of early rumor detection method

Original Deep learning Module: In the upper part of Fig. 1, from an original deep learning rumor detection model M, we decompose a sub model M_f, which consists of the input layer and all the hidden layers. With the inputs of source post P_0 and the responses P_1, P_2, ..., and P_N, the feature f is obtained through the hidden layers. The output layer is often a linear layer with a sigmoid function.

CGAN module: In the middle part of Fig. 1, the connection between the source post P_0 and the feature f is established through a conditional generative adversarial network (CGAN). The generator outputs the fake feature f' based on the input of P_0 and a noise parameter, while the discriminator tries to judge whether f and f' are real. The symbol $\oplus$ indicates the concatenation of two vectors. After the adversarial training of CGAN, the generator is finally able to generate the features containing the information of responses.

MLP module: In the lower part of Fig. 1, M_f first extracts the feature f from the inputs of P_0, P_1, P_2, ..., and P_N, meanwhile the generator outputs a fake feature f' by the source post P_0, and finally f and f' are synthesized and fed to a MLP (multi-layer perception) to get the rumor detection results.

B. Design of CGAN

The CGAN module includes a generator and a discriminator. The generator concatenates the source post P_0 and the noise as the input, and outputs the generated feature by a three layered fully-connected network,

$$\begin{aligned} x_0 &= (z, P_0) \\ x_i &= relu(w_{i-1}x_{i-1} + b_{i-1}), i = 1,2 \\ f &= \tanh(w_2 x_2 + b_2) \end{aligned} \tag{1}$$

The discriminator is another three layered perceptron, which concatenates the source post and the feature as the input, and then outputs the probability of the feature being real or generated,

$$\begin{aligned} x_3 &= (f, P_0) \\ x_i &= relu(w_{i-1}x_{i-1} + b_{i-1}), i = 4,5 \\ r &= w_5 x_5 + b_5 \end{aligned} \tag{2}$$

Let θ be the parameters (w_0, w_1, w_2, b_0, b_1, b_2), w be the parameters (w_3, w_4, w_5, b_3, b_4, b_5), then the generator and discriminator are denoted by $f = G_\theta(z, P_0)$ and $r = D_w(x, P_0)$ respectively. For the convenience, we use superscript to indicate the id of the event, such as $E^i = (P_0^i, P_1^i, \cdots, P_N^i)$. Let M_f denote the sub model decomposed from the original deep learning model, we use Wasserstein Loss [25] in the training of the discriminator,

$$J_w = \frac{1}{m}\sum_{i=1}^{m}\left(D_w\left(M_f\left(E^i\right), P_0^i\right)\right) - \frac{1}{m}\sum_{i=1}^{m}\left(D_w\left(G_\theta(z^i, P_0^i), P_0^i\right)\right) \tag{3}$$

where m is the batch size of the data. The loss function of the generator is defined as (4),

$$J_\theta = -\frac{1}{m}\sum_{i=1}^{m}\left(D_w\left(G_\theta(z^i, P_0^i), P_0^i\right)\right) \tag{4}$$

C. Design of MLP

The final process of the early detection is finished by a two layered MLP, which concatenates the real feature $f = M_f(E)$ and the generated feature $f' = G_\theta(z, P_0)$ as the input, and then outputs the probability of the event being rumor,

$$\begin{aligned} x_6 &= (M_f(E), G_\theta(z, P_0)) \\ x_7 &= relu(w_6 x_6 + b_6) \\ p &= sigmoid(w_7 x_7 + b_7) \end{aligned} \tag{5}$$

We use the cross-entropy loss to train the MLP,

$$J = -\frac{1}{m}\sum_{i=1}^{m}\log(p_i)y_i + (1-\log(p_i))(1-y_i)) \tag{6}$$

where y_i is the label of the event E^i.

V. Experiments

A. Three Deep Learning Models to Be Improved

We use the proposed method to improve three recent reported deep learning rumor detection models with different early detection ability. The sub model decompositions are illustrated in Fig 2, where the layers before "feature" make the sub model M_f.

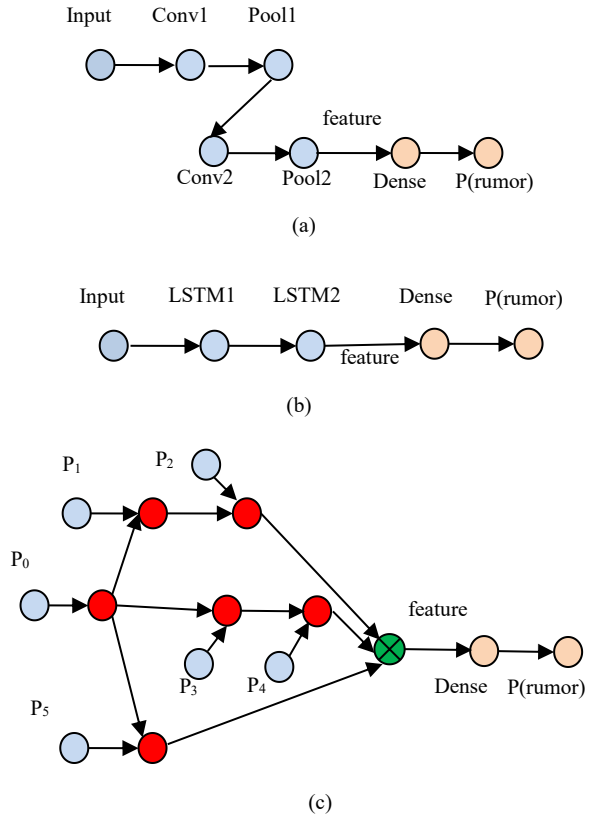


Figure 2. The models of CNN(a), LSTM(b), and RvNN(c)

CNN model [3]. As in Fig 2-a, in the input layer, the authors divided all the posts of an event into groups by time and then used doc vector to represent each group. They used two convolutional lays with two pooling layers as the hidden

layers to extract the feature, and a fully-connected layer as the output layer to make the classification.

LSTM model [4]. As in Fig 2-b, the authors also divided the input posts into groups by time and then got a data sequence. After converting the data into vectors, they extracted the feature by two LSTM layers. Finally, they used a fully-connected layer with sigmoid activation to finish the rumor detection.

RvNN model [5]. As in Fig 2-c, the authors found that the propagation of the posts in an event is tree-structured, so they configured an RvNN layer at each node (post) of the tree. Taking the post in a node and the output of its parent node as the input, RvNN layer makes a computation and the results flow to the child nodes. The computation is recursively continued till to each leaf node, and then the outputs of all leaf nodes are pooled to get the feature. The classification is performed by a fully-connected layer finally.

In comparison, the CNN and LSTM models [3, 4] group all the posts and then extract the features on the whole data, while the RvNN model [5] processes each post one by one, so the RvNN model depends more on the pattern of single post and thus it has better early detection ability than the CNN and RNN models.

B. Dataset

The experiments were on two large Sina Weibo datasets, and their statistical information is shown in TABLE I. .

TABLE I. WEIBO AND CHECKED DATASETS.

Name	Weibo	CHECKED
Year	2016	2021
Number of rumor	2313	344
Number of non-rumor	2351	1760
Average # of responses in a rumor	741	48
Average # of responses in a non-rumor	889	664

The first dataset Weibo is a large Sina Weibo dataset reported in the paper [4], with events covering various aspects of politics, economics, entertainment, etc. It is a balanced dataset and each event in it is with a large number of responses, the complete propagation information is also provided.

The second dataset CHECKED is a Sina Weibo dataset recently published in the paper [26], which is special about the COVID-19 events. It is unbalanced dataset with a ratio of positive and negative samples of 1:5. In addition, the average numbers of responses of rumor and non-rumor are very different, that the former is only 48. It should be because that the society is sensitive to COVID-19 rumors that they were ended in short time.

The improvements of the CNN, LSTM, and RvNN were tested on the Weibo dataset. Because the CHECKED dataset does not provide the tree structure information among the responses, so the RvNN was not test on it.

C. Parameter Setting

In the experiments, the same hyper-parameters are used for the CGAN which can be modified according to the models to be improved in practice. For the generator, P_0 needs to be normalized and the noise is standard normally distributed, and they are both 100-dimensional vectors. After the concatenation, a 200-dimensional vector is obtained and then fed to three fully connected layers and the dimensions become 160, 120 and 100 in turns. Because there are positive and negative elements in the real feature vector, we use *tanh* as the active function in the last layer. For the discriminator, the input P_0 and features are both 100-dimensional vectors and concatenated into a 200-dimensional vector, and then it is fed to a three layer perceptron, the dimension is convert to 100, 20, and 1 in turns. We choose RMSprop [24] as the optimizer. The learning rate α is set to 5×10^{-5}, the truncation amplitude c is set to 0.01, the batch size m is set to 128, and the discriminator-generator training ratio n is set to 5.

The input layer of the MLP module concatenates two 100-dimension feature vectors and gets a 200-dimensioal vector. Passing through two fully-connected layers, the dimensions of the vector become 100 and 1 in turn, and with a sigmoid activation, we get the probability of being a rumor. The optimizer of MLP is Adagrad, and the learning rate is set to 0.01。

D. Results and Analysis

The 5-fold cross-validation was performed for each model. In the prediction step, we chose k responses of each event in the time order to test early detection performance of the original models and the proposed model. The accuracy, precision, recall, and F1 are used as the metrics of evaluation. In addition, due to the unbalance in CHECKED, Macro F1 is also used.

1) Performance of the original models on all responses

We first trained the three original models with all the responses. Because CGAN is trained on the features extracted by the original models, their performances are vital to the proposed method. The results in TABLE II show that the trained original models all have good performance, and provide a good basis for the subsequent steps.

TABLE II. MACRO F1 VALUE OF ORIGINAL MODELS.

Dataset	CNN	LSTM	RvNN
Weibo	0.930	0.933	0.912
CHECKED	0.985	0.982	\

2) Results on Weibo

a) Early detection with zero response

TABLE III shows the performance of the proposed method in early detections when there is zero response, where CNN, LSTM and RvNN are the original models and the iCNN, iLSTM and iRvNN are the corresponding improved models.

We find that if only the source post is provided, the performances of three original models are seriously degraded compared to the results inTABLE II. It shows that the responses play an important role in the detection. The accuracies of CNN and LSTM are just 0.504 and 0.582. We noted that the recall of class rumor is quite small which means a large number of rumors are misclassified to be non-rumors. By the help of the generated features, the accuracies of iCNN

and iLSTM are increased about 23% and 13% respectively. The macro F1 of them are also improved about 37% and 20% respectively.

Among the three models, the original RvNN has better performance when there is no response that it gets an accuracy of 0.734. The improved model iCNN has already outperformed the RvNN model in accuracy, while iLSTM is closed to it. Moreover, the RvNN can also be improved that the accuracy and Macro F1 of iRvNN are about 1% above the original RvNN.

TABLE III. RESULTS ON WEIBO WITH ZERO RESPONSE

Models	Acc	Macro F_1	Rumor			Non-rumor		
			Prec	Recall	F_1	Prec	Recall	F_1
CNN	0.504	0.335	0	0	0	0.504	1.000	0.670
iCNN	0.737	0.718	0.921	0.494	0.643	0.673	0.961	0.792
LSTM	0.582	0.505	0.862	0.19	0.309	0.549	0.968	0.700
iLSTM	0.716	0.702	0.823	0.519	0.637	0.669	0.897	0.767
RvNN	0.734	0.729	0.689	0.860	0.765	0.810	0.606	0.693
iRvNN	0.743	0.743	0.688	0.926	0.789	0.871	0.544	0.699

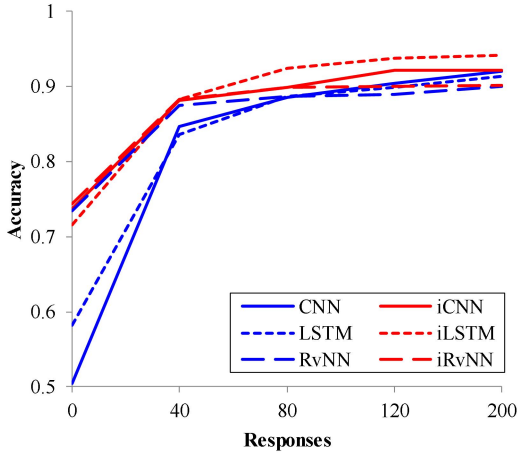


Figure 3. Results on Weibo with some responses

b) Early Detection in different time stages

With the increase of the responses, the accuracies of the three original models increase. However, the proposed method can still improve their performance in certain time stages (Fig. 3). The iCNN model improves the accuracy about 3.4% when there are 40 responses, and it keeps the superiority until there are 200 responses. The iLSTM model improves the accuracy about 4.7% in the first 40 responses, and the improvement is kept above 3% till 200 responses. For the iRvNN model, its improvement is about 1% over the RvNN until there are 200 responses. Among the six models, iLSTM gains the best early detection performance when there are some responses.

According to TABLE I, the average response in each event is less than 900, which indicates that the proposed method can be applied in a long early time stage.

3) Results on CHECKED

The results on the CHECKED dataset are shown in TABLE IV and Fig. 4. Because the responses in rumor class are relatively few, we chose 80 responses at most to test the early detection performance. The proposed method also shows the performance superiority.

In the cases of zero responses, CNN and LSTM have bad accuracy. The precision and recall show that the models tend to wrongly classify most events as rumors. The accuracies of the proposed iCNN and iLSTM are improved above 60% in the case of zero response.

When there are some responses, the accuracies of CNN and LSTM are both very high. However iCNN and iLSTM can still improve the accuracy about 1% and the superiorities are kept in the early 80 responses.

TABLE IV. RESULTS ON CHECKED WITH ZERO RESPONSE

Models	Acc	Macro F_1	Rumor			Non-rumor		
			Prec	Recall	F_1	Prec	Recall	F_1
CNN	0.164	0.142	0.164	1.000	0.281	0.400	0.001	0.002
iCNN	0.905	0.840	0.622	0.903	0.737	0.982	0.905	0.942
LSTM	0.163	0.141	0.163	1.000	0.281	0	0	0
iLSTM	0.845	0.775	0.488	0.967	0.649	0.993	0.824	0.901

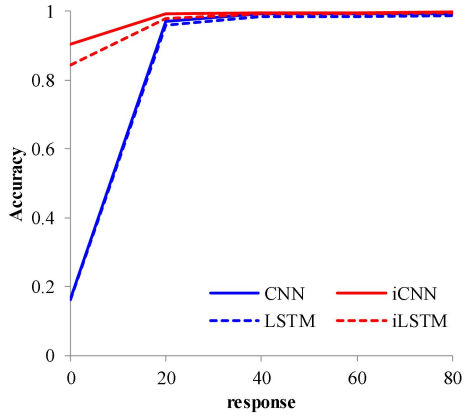


Figure 4. Results on CHECKED with some responses

4) Cases Study

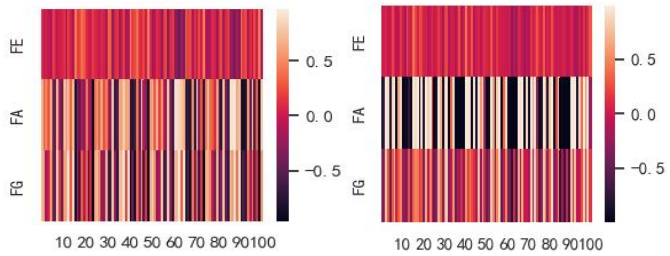
Because the generated features are just vectors of real number, we cannot observe the semantic information directly from them. However, the relationship between the feature generated from the source post (FG), the feature extracted by the original model from some responses in the early stage (FE), and the features extracted by the original model from all the responses (FA) can be investigated through data visualization.

Fig. 5-a is from the event with ID 3514388935498432 in Weibo dataset, which is a rumor correctly predicted by the iCNN model with zero response, but the original CNN model makes wrong prediction. In the figure, the x-axis represents the elements of the 100-dimensinal feature vector, and the color represents the value of the each element. We can see that the magnitude and variation of the elements in the generated feature (FG) is more similar to the feature from all the responses (FA) compared to the early feature (FE) extracted by CNN.

Fig. 5-b shows the different situation of another event with ID 3912024620676243, which is a non-rumor, and the

prediction is on 40 responses. The CNN model classifies it as a rumor incorrectly, but the iCNN model makes a correct prediction. It can be seen that the variation of the elements in FG is still more similar to the FA compared to the FE of the original CNN model.

The case study shows that the CGAN model is able to simulate the distribution of the responses.



(b)

Figure 5. Comparison of the features of FG, FE, and FA from two sample events in Weibo.

VI. CONCLUSIONS

In the early stage of rumor propagation, the performance of existing deep learning models is not high due to the small amount of response data. In this paper, we generate the feature data containing the response information based on the source post by the conditional adversarial generative network, and the generated feature is combined with the real feature to improve the early detection performance. The effectiveness and generality of this method are verified on three deep learning models.

Future work is to apply this method to more deep learning models, and investigate how to make the improvements more effective.

REFERENCES

[1] Y. Liu, "Two Defendants Were Sentenced in Yili Rumor Case ", ed: China News Agency 2018.

[2] G. Liang, W. He, C. Xu, and L. Chen, "Rumor Identification in Microblogging Systems Based on Users Behavior," *IEEE Transactions on Computational Social Systems,* vol. 2, no. 3, p. 10, 2015.

[3] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A Convolutional Approach for Misinformation Identification," presented at the Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.

[4] J. Ma *et al.*, "Detecting Rumors from Microblogs with Recurrent Neural Networks," presented at the IJCAI2016 New York, USA, July 9–15, 2016, 2016.

[5] J. Ma, W. Gao, and K.-F. Wong, "Rumor Detection on Twitter with Tree-structured Recursive Neural Networks," presented at the Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 2018.

[6] J. Ma, W. Gao, S. Joty, and K. F. Wong, "An Attention-based Rumor Detection Model with Tree-structured Recursive Neural Networks," *ACM Transactions on Intelligent Systems and Technology,* vol. 11, no. 4, pp. 1-28, 2020.

[7] T. Chen, X. Li, H. Yin, and J. Zhang, "Call Attention to Rumors: Deep Attention Based Recurrent Neural Networks for Early Rumor Detection," presented at the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[9] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Eprint Arxiv,* 2014.

[10] Z. Liu, Z. Wei, and R. Zhang, "Rumor Detection Based on Convolutional Neural Network," *Journal of Computer Applications,* vol. 37, no. 11, pp. 3053-3056, 2017.

[11] Y. Xu, C. Wang, Z. Dan, S. Sun, and F. Dong, "Deep Recurrent Neural Network and Data Filtering for Rumor Detection on Sina Weibo," *Symmetry,* vol. 11, no. 11, p. 1408, 2019.

[12] T. N. Nguyen, C. Li, and C. Nieder´ee, "On Early-stage Debunking Rumors on Twitter: Leveraging the Wisdom of Weak Learners," presented at the The 9th International Conference on Social Informatics, 2017.

[13] Y. Liu and Y.-F. Wu, "Early Detection of Fake News on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks," presented at the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[14] Y.-Z. Song, Y.-S. Chen, Y.-T. Chang, S.-Y. Weng, and H.-H. Shuai, "Adversary-Aware Rumor Detection," presented at the ACL-IJCNLP 2021, 2021.

[15] Tian Bian *et al.*, "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks," presented at the AAAI 2020, 2020.

[16] L. Wei, D. Hu, W. Zhou, Z. Yue, and S. Hu, "Towards Propagation Uncertainty: Edge-enhanced Bayesian Graph Convolutional Networks for Rumor Detection," presented at the ACL2021, 2021.

[17] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems,* vol. 3, pp. 2672-2680, 2014.

[18] D. Saxena and J. Cao, "Generative Adversarial Networks (GANs)," *ACM Computing Surveys (CSUR),* vol. 54, pp. 1 - 42, 2021.

[19] D. Chang, W. Yang, X. Yong, G. Zhang, and Y. Wang, "Seismic Data Interpolation Using Dual-Domain Conditional Generative Adversarial Networks," *IEEE Geoscience and Remote Sensing Letters,* vol. PP, no. 99, pp. 1-5, 2020.

[20] H Zhang *et al.*, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," *2017 IEEE International Conference on Computer Vision (ICCV),* 2017.

[21] S. W. Huang, C. T. Lin, S. P. Chen, Y. Y. Wu, and S. H. Lai, "AugGAN: Cross Domain Adaptation with GAN-based Data Augmentation," presented at the ECCV, 2018.

[22] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camera Style Adaptation for Person Re-identification," presented at the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18-23 June 2018, 2018.

[23] Y. Wang *et al.*, "EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection," presented at the 24 th Acm Sigkdd International Conference, 2018.

[24] J. Ma, W. Gao, and K.-F. Wong, "Detect Rumors on Twitter by Promoting Information Campaigns with Generative Adversarial Learning," presented at the The World Wide Web Conference, San Francisco, CA, USA, 2019.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN." doi: arXiv:1701.07875 [stat.ML]

[26] C. Yang, X. Zhou, and R. Zafarani, "CHECKED: Chinese COVID-19 Fake News Dataset," *Social Network Analysis and Mining,* vol. 11, no. 1, p. 58, 2021.

Recurrent Graph Convolutional Network for Rumor Detection

Song Wu, Hailing Xiong*, Ye Yang, Jinming Zhang, Chenwei Lin
College of Computer and Information Science, Southwest University, Chongqing, China
wusong5543@gmail.com

***Abstract*—The development of social media has provided an ideal platform for sharing information, but it has also become a hotbed for rumor posting and spreading. Existing rumor detection methods mainly look for clues from the textual content, static propagation structures. However, existing studies do not make full use of user information while ignoring the dynamic change of the communication structure. Therefore, this paper proposes a rumor detection model based on dynamic propagation structure called Recurrent Graph Convolutional Network (Re-GCN). In terms of rumor content representation, both the textual content of rumors and user information are considered. In terms of propagation structure, the dynamic propagation structure of rumors is considered. A dynamically changing propagation structure representation is constructed by dividing rumor propagation into multiple stages, and the dynamically changing propagation features are captured using a Bi-directional Recurrent Neural Network (Bi-RNN) and Graph Convolutional Networks (GCN). Finally, the attention mechanism is introduced to fuse the information of each stage to classify rumors. Experimental results on two real-world datasets show that the proposed method has significant improvement compared to the state-of-the-art benchmark methods.**

Keywords-rumor detection; graph convolutional network; dynamic propagation structure; social media

I. INTRODUCTION

Mobile Internet has become an integral part of people's lives. The openness and convenience of social media platforms such as Twitter, Facebook, and Weibo have lowered the threshold of information exchange, but it has also become a hotbed for rumors to spread. An official report released by Weibo shows that the Weibo platform effectively dealt with 76,107 rumors in 2020[1]. Rumors seriously mislead people's minds and even cause substantial economic losses, and endanger public safety. Therefore, there is an urgent need for fast and effective methods to detect social media rumors in response to the potential threats that rumors may pose.

Most early rumor detection methods mainly used manually extracted features such as text content[2] [3], user features[2], and propagation methods [4] [5] to train supervised classifiers, e.g., decision trees[2], support vector machine (SVM)[6]. Some studies have mined more effective features, such as temporal structure features[7], sentiment attitude of posts[8]. However, such methods rely heavily on feature engineering, which is time-consuming and suffers from insufficient robustness.

Recent studies have mainly employed deep learning methods to fully exploit the deep features of rumors by learning high-level representations of rumor texts through various neural networks, such as RNN and Gated Recurrent Unit(GRU)[10], to detect rumors. Some studies also considered the propagation structure of rumors and constructed top-down propagation trees and bottom-up propagation trees based on the retweet-reply relationships among posts in social media, as shown in Fig.1. The node "1" represents the source post of the rumor, and the other nodes represent the retweeted messages related to the source post. Then, the structural features of rumor propagation are captured by Recursive Neural Networks(RvNN)[11] and GCN[12][15].

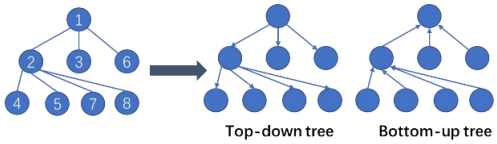


Figure 1. A bottom-up/top-down propagation tree. Node "1" represents the source post, i.e., the earliest post, and the other nodes represent the re-posted comment messages related to the source post. The number in the node indicates the order in which the messages are posted, and the larger the number, the later the message is posted.

Although the above studies have achieved good results in rumor detection tasks, there are still two shortcomings. The first is in the representation of the content of the rumor, these methods mainly consider rumor text content and ignore the user information contained in social media, which proved to be an essential clue for rumor detection in earlier studies[2]. The second is in the representation of the structure of rumor propagation, they consider the static structure of rumor propagation and ignore the change of rumor propagation structure over time, i.e., the dynamic propagation structure. Taking Fig.2 as an example, the global propagation structures of the propagation trees (a) and (b) are the same, but there are differences in their propagation structures when different cutoff times are chosen.

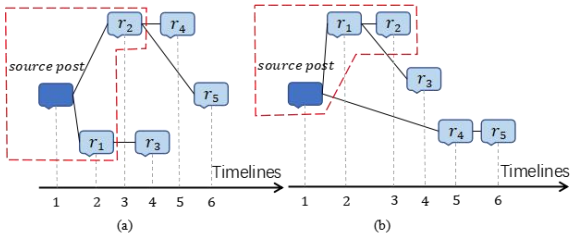


Figure 2. When the selected cutoff time is "6", which is the case of global propagation structure, the propagation trees (a) and (b) have the same structure. However, in the case of not global propagation structure, for example, when the cutoff time is "3", the structure of (a) can be described as "$s(source\ post) \rightarrow r_1, s \rightarrow r_2$" and the structure of (b) can be described as "$s \rightarrow r_1 \rightarrow r_2$ ", which are different.

DOI reference number: 10.18293/SEKE2022-063

To address the above issues, our research is concerned with (1) Considering both textual content and user information in the rumor detection task. (2) modeling the dynamic propagation structure of rumors and applying it to rumor detection. In this paper, we propose a novel Recurrent Graph Convolutional Network (Re-GCN) for rumor detection. First, we consider both textual content and user information during rumor propagation and extract features from them as the content representation of rumors. Second, we constructed a representation of the dynamic propagation structure by dividing the propagation tree into multiple stages at the same number interval, and Fig.3 shows an example of dividing a rumor propagation tree into 4 stages. We also propose the Re-GCN model to model the dynamic propagation structure and classify the rumors. The main contributions of this work are as follows.

- We consider both textual content and user information in social media, and encode them separately using different representations.
- We analyze and introduce the dynamic propagation structure of rumors. A multi-stage propagation structure representation method is proposed and a Recurrent Graph Convolutional Network (Re-GCN) is designed to model the dynamic propagation structure.
- Experiments on the Weibo and PHEME datasets show that our model improves significantly on the rumor detection task compared to some state-of-the-art models.

The rest of this paper is organized as follows. In Section II, we describe the related work on rumor detection. The problem statement is discussed in Section III. In Section IV, we describe the details of the proposed Re-GCN model. In Section V, we present the experiments and analyze the experimental results. In Section VI, we summarize the research of this paper.

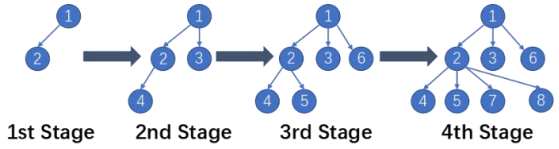


Figure 3. The rumor propagation is divided into 4 stages based on the number of nodes interval to represent the dynamic propagation of rumors, and the difference in the number of nodes between each stage is the same.

II. Related Work

In recent years, research on automatic rumor detection on social media has attracted much attention. For the sake of description, this paper adopts the concept of "event", which consists of a source post and retweeted posts related to the source post. The rumor detection task aims to determine whether the event is a rumor based on information of the source post and related retweeted comments. Related research can be divided into two main categories: (1) Feature engineering-based approaches; (2) Deep learning-based approaches.

Most early studies mainly used feature engineering-based approaches to extract hand-crafted features from event-related post messages and classify events using classifiers such as decision trees, random forests, and SVM. Castillo et al. [2] considered features such as the average sentiment score of messages, and the proportion of messages with URLs in their research work on Twitter news topic credibility assessment. Yang et al.[6] considered two new features in their work on rumor detection: the client used by the user to post the message and the location of the event mentioned in the message content; Kwon et al.[16] proposed a method for rumor detection based on rumor propagation time, structure, and linguistic features; Zhang et al.[17] considered four implicit content-based features: popularity, internal and external consistency, sentiment polarity and comment opinion. Feature engineering-based methods usually require extensive pre-processing and feature design processes, which are less efficient. Hand-crafted features also have the limitation of not being robust enough.

As deep learning methods have achieved remarkable results in various natural language processing tasks, many scholars have also applied deep learning models to rumor detection. Ma et al.[10] first applied deep learning models to the field of rumor detection, using RNN and GRU to model rumor propagation, and achieved significant improvements compared to previous benchmark methods. Chen et al.[18] improved the RNN-based approach by combining attentional mechanisms with RNN and introducing attentional mechanisms to capture text features. Yu et al.[9] used Doc2vec method to obtain text vectors of tweets in periods, stitched the text vectors of each period into a feature matrix of events, and used CNN to learn the hidden layer representation of events. Bian et al.[12] first applied GCN to rumor detection research and proposed a Bi-Directional Graph Convolutional Neural Network(Bi-GCN) model that takes into account the top-down and bottom-up propagation structure of rumors. Wei et al.[13] considered the propagation uncertainty of rumor detection and proposed an Edge-enhanced Bayesian Graph Convolutional Network(EBGCN) to capture robust structural features. Lao et al.[14] considered both linear temporal sequence and the non-linear diffusion structure, capturing linear sequence features by LSTM and nonlinear structural features by GCN. For the limited rumor labeling data, He et al.[15] designed three event enhancement strategies and proposed the Rumor Detection on social media with Event Augmentations(REDA) framework to learn event representation by self-supervised pre-training. However, most of these methods focus on the textual content of rumors and ignore other types of information in social media. In contrast, the representation of the rumor propagation structure mainly focuses on the static structure of rumor propagation and ignores the dynamic change of the rumor propagation structure with the sequential posting of posts.

Previous work has provided valuable ideas for our study. To address the deficiencies in content representation, we consider both textual content and user information. In terms of propagation structure, we introduce a dynamic propagation structure representation and design a Re-GCN model for rumor detection.

III. Problem Statement

In this paper, the rumor detection dataset is defined as $C = \{c_1, c_2, ..., c_m\}$where c_i represents the i-th event and m is the number of events. The ground-truth label of all events defined as $Y = \{y_1, y_2, ..., y_m\}$ where y_i denotes the label of c_i. c_i contains several posts, $c_i = \{r_i, w_1^i, w_2^i, ..., w_{n_i-1}^i, \}$,where n_i denotes the number of posts contained in c_i, r_i denotes the

source post, and w_j^i denotes the j-th related reply or retweet post. We denote the propagation structure of c_i by the directed graph $G_i =< V_i, E_i >$. where V_i denotes the set of nodes of G_i and E_i denotes the set of edges of G_i. As shown in Fig.1, each post is a node, and if the post w_1^i replies to r_i, there is a directed edge from r_i to w_1^i.

The problem to be solved in this paper can be summarized as follows: given a rumor dataset, learn a classifier f that maps each event in C to the ground-truth label(rumor/non-rumor)

$$f: C \rightarrow Y.$$

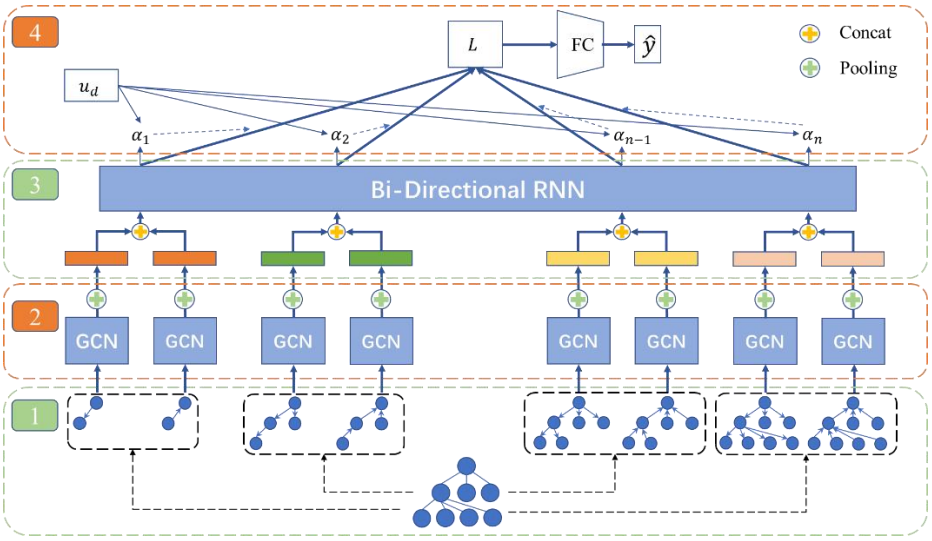


Figure 4. Our Re-GCN rumor detection model.

IV. RE-GCN RUMOR DETECTION MODEL

This section proposes a dynamic propagation structure-based rumor detection method called Recurrent Graph Convolutional Network (Re-GCN). The core idea of Re-GCN is to learn the appropriate high-level representation from the propagation structure of rumors at each stage. Re-GCN consists of four components: post content representation module, GCN module, Bi-RNN module, feature fusion and classification module. Specifically, the event representation module describes mapping the post text and user information in the event to the vector space and the construction process of the dynamic propagation structure representation of the event. The GCN module obtains the high-level node representation of each stage of event propagation by GCN. The Bi-RNN module models the dynamic changes of the propagation structure by using the high-level node representations of the event phases obtained from the GCN module as the input sequence of the Bi-RNN. The feature fusion and rumor detection module use the attention mechanism to fuse the high-level features of the events in each stage and perform classification. Fig.4 shows the structure of the proposed model.

We discuss how to use the Re-GCN to detect an event c with post count n. The same calculation is used for all other events.

A. Event Representation

The event representation of rumors is mainly divided into post content representation and dynamic propagation structure representation. We arrange all posts in the event according to the posting time from early to late, dividing them into N copies according to the number equally, which corresponds to the N stages of rumor propagation, and N takes the value of 6 in this paper. The propagation structure of the t-th stage contains the first t number of posts, i.e., $\frac{tn}{N}$ posts.

1) Post Content Representation

For the post content, taking a post in an event as an example, we consider both the user information and the text content of the post. User information features mainly refer to the content of the user account registration and some statistical features under that account. Table I and Table II show the user information in the selected Weibo[10] and PHEME[19] datasets.

TABLE I. USER INFORMATION EXTRACTED FROM WEIBO.

No.	Feature	Type
1	GENDER	Binary
2	VERIFIED	Binary
3	USER_GEO_ENABLED	Binary
4	BI_FOLLOWERS_COUNT	Integer
5	FRIENDS_COUNT	Integer
6	FOLLOWERS_COUNT	Integer
7	STATUSES_COUNT	Integer
8	FAVOURITES_COUNT	Integer
9	COMMENTS_COUNT	Integer

TABLE II. USER INFORMATION EXTRACTED FROM PHEME.

No.	Feature	Type
1	VERIFIED	Binary
2	USER_GEO_ENABLED	Binary
3	FAVOURITES_COUNT	Integer
4	RETWEET_COUNT	Integer
5	FOLLOWERS_COUNT	Integer
6	LISTED_COUNT	Integer
7	STATUSES_COUNT	Integer
8	FRIENDS_COUNT	Integer

Then we describe the preprocessing of these features. We preprocess them using One-Hot encoding for binary features such as "VERIFIED" in user information. For integer features such as "FRIENDS_COUNT" in user information, we use normalization to process them. The discrete features are then spliced with the numerical features as the user information feature representation. For text content, BERT[20] is an excellent algorithm to obtain text vector representation, and we use the pre-trained BERT-base model to extract the representation vector of text content. Finally, the user information feature representation is stitched with the text content representation to obtain the post content representation vector.

We construct the content feature matrix $X = \{X_1, X_2, ..., X_N\}$ for each stage of the event based on the N stages of rumor propagation, where X_t denotes the content feature matrix of the t-th stage, which consists of the content representation vectors of the first $\frac{tn}{N}$ posts.

2) Dynamic propagation structure representation

Taking stage t of rumor propagation as an example, we define the adjacency matrix A_t corresponding to the top-down propagation structure based on the forward-reply relationship between posts in the t stage. The values of the adjacency matrix A_t are defined in Eq.1. We consider both the top-down and bottom-up propagation structures, and the adjacency matrix corresponding to the top-down propagation structure is denoted as $A_t^{TD} = A_t$. The adjacency matrix corresponding to the bottom-up propagation structure is denoted as $A_t^{BU} = A_t^T$, A_t^{BU} is the transpose of A_t.

$$(A_t)_{ij} = \begin{cases} 1, & if\ tweet\ i\ replies\ to\ tweet\ j \\ 0, & otherwise \end{cases} \quad (1)$$

We construct the adjacency matrix $A^{TD} = \{A_1^{TD}, A_2^{TD}, ..., A_N^{TD}\}$ and $A^{BU} = \{A_1^{BU}, A_2^{BU}, ..., A_N^{BU}\}$ for both top-down and bottom-up propagation structures of N stages.

B. GCN module

GNN-based methods have achieved impressive results in many fields (e.g., click-through rate prediction[22], text classification[23]), among which GCN is a very effective method for processing graph structured data by updating the node's embedding based on its neighbors. GCN can capture information from direct and indirect neighbors of a node through a stacked hierarchy. There exist several types of message propagation functions for GCN. In this paper, we use the message propagation function defined by GCN in the first-order approximation of ChebNet[21]. For a top-down propagation structure, the 2-layers GCN is calculated as

$$H_{t_0}^{TD} = \sigma\left(\widehat{A}_t^{TD} X_t W_{t_0}^{TD}\right) \quad (2)$$

$$H_{t_1}^{TD} = \sigma\left(\widehat{A}_t^{TD} H_{t_0}^{TD} W_{t_1}^{TD}\right) \quad (3)$$

Where $\widehat{A}_t^{TD} = D^{-\frac{1}{2}} \widetilde{A}_t^{TD} D^{-\frac{1}{2}}$ is the normalized adjacency matrix, $\widetilde{A}_t^{TD}$ is A_t^{TD} plus the self-loop, D is the degree matrix, and $W_{t_0}^{TD}$ and $W_{t_1}^{TD}$ denote the GCN weight parameters of the first and second layers, respectively. For the bottom-up propagation structure, the same computational procedure can be used to obtain $H_{t_1}^{BU}$. We use $ReLU$ as the activation function. To prevent overfitting, we apply the Dropout[24] regularization method to the GCN.

We use mean pooling to aggregate information from these two sets of node representations and then stitch the top-down propagation features and bottom-up propagation features to obtain the final representation of the propagation features. It is formulated as

$$S_t^{TD} = MEAN\left(H_{t_1}^{TD}\right) \quad (4)$$

$$S_t^{BU} = MEAN\left(H_{t_1}^{BU}\right) \quad (5)$$

$$S_t = concat(S_t^{TD}, S_t^{BU}) \quad (6)$$

For each of the N stages of rumor propagation, we obtain its propagation characteristics, which are denoted as $S = \{S_1, S_2, \ldots, S_N\}$ for all stages.

C. Bi-RNN module

RNN-based approaches are effective in modeling sequential data. We use Bi-RNN to model the dynamically changing propagation structure with S as the input sequence of Bi-RNN. Bi-RNN consists of forward RNN and backward RNN, and the computation procedures of forward RNN and backward RNN are shown in Eq. 7and Eq. 8, respectively. Then we splice the output of forwarding RNN and backward RNN as the output of the current moment, as shown in Eq. 9.

$$\overrightarrow{h_t} = \tanh\left(\overrightarrow{U} S_t + \overrightarrow{W} \overrightarrow{h}_{t-1} + \overrightarrow{b}\right) \quad (7)$$

$$\overleftarrow{h_t} = \tanh\left(\overleftarrow{U} S_t + \overleftarrow{W} \overleftarrow{h}_{t+1} + \overleftarrow{b}\right) \quad (8)$$

$$h_t = concat\left(\overrightarrow{h_t}, \overleftarrow{h_t}\right) \quad (9)$$

Where $(\overrightarrow{U}, \overrightarrow{W})$ and $(\overleftarrow{U}, \overleftarrow{W})$ are the weight parameters inside the forward RNN($\overrightarrow{RNN}$) and backward RNN($\overleftarrow{RNN}$) hidden layer cells, $\overrightarrow{b}$ and $\overleftarrow{b}$ are the bias of the $\overrightarrow{RNN}$ and $\overleftarrow{RNN}$. S_t denotes the propagation characteristics of the event at stage t, tanh is the activation functions of $\overrightarrow{RNN}$ and $\overleftarrow{RNN}$, respectively, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the outputs of the $\overrightarrow{RNN}$ and $\overleftarrow{RNN}$ at moment t, and h_t denotes the splicing of $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ as the outputs of the Bi-RNN at moment t.

D. Feature fusion and classification

In order to enable the model to make focused use of the information in each stage of rumor propagation, we use the attention mechanism to assign attention weights to the output of the hidden state by the Bi-RNN at each moment. The attention mechanism is calculated as

$$o_t = \tanh\left(W_d h_t + b_d\right) \quad (10)$$

$$\alpha_t = \frac{\exp\left(o_t{}^T u_d\right)}{\sum_t \exp\left(o_t{}^T u_d\right)} \quad (11)$$

$$L = \sum_t \alpha_t o_t \quad (12)$$

Where o_t is the hidden layer representation of h_t after a hidden layer with $tanh$ as the activation function, W_d and b_d are the weight parameter matrix and bias, respectively. u_d is the random initialized weight parameter, α_t is the calculated weight value of the hidden state at each moment of the Bi-RNN, and L is the sequence representation obtained

by weighting and summing the hidden state values at each moment. Finally, a fully connected layer and a $softmax$ layer are used to calculate the prediction results $\hat{y}$ of the events.

$$\hat{y} = Softmax\big(FC(L)\big) \tag{13}$$

We train all parameters of the Re-GCN model by minimizing the cross-entropy of the ground-truth labels and predicted labels. To speed up the convergence of the model, we use the Adam[25] optimization algorithm to update the model parameters.

V. EXPERIMENTS

In this section, we compare the performance of the proposed Re-GCN model with that of some state-of-the-art benchmark models.

A. Datasets

We evaluate our proposed method on two real-world datasets: Weibo[10] and PHEME[19]. In all datasets, nodes refer to posts, and edges represent retweet or reply relationships. Both Weibo and PHEME contain two binary labels, false rumors (F) and true rumors (T). The tags of each event in Weibo are annotated by the Sina Community Management Center, which reports various error messages[10]. PHEME contains relevant rumor and non-rumor posts that appeared on Twitter during the period when some breaking news was released. The statistics for both datasets are shown in Table III.

TABLE III. STATISTICS OF THE DATASETS

Statistic	PHEME	Weibo
Of source tweets	5802	4664
Of tree nodes	30376	3805656
Of non-rumors	3803	2351
Of rumors	1972	2313
Avg. time length/tree	18 Hours	2461 Hours
Avg. of posts/tree	6	816
Max of posts/tree	228	59318
Min of posts/tree	3	10

B. Baselines

We compare the proposed model with some state-of-the-art benchmarking methods, including feature engineering-based methods and deep learning-based methods.

The comparison methods are as follows. DTC[2]: a rumor detection method that uses a decision tree classifier based on various manual features to obtain the credibility of information. SVM-TS[26]: a linear SVM classifier that uses hand-crafted features to construct a time series model for rumor detection. DTR[27]: a decision tree based ranking model for rumor detection by querying phrases. GRU[10]: a recurrent neural network-based rumor detection model with GRU units that learns rumor representations by modeling the sequential structure of related posts. RvNN[11]: a rumor detection method based on tree-structured recurrent neural networks with GRU units that learn rumor representation by propagation structure. Bi-GCN[12]: a GCN-based rumor detection model that captures the features of rumor propagation through top-down and bottom-up rumor propagation structures.

As in the original paper[12], we randomly partitioned the dataset into five sections and performed 5-fold cross-validation to obtain stable results. For Bi-GCN, which also uses graph structures, we conducted experiments using the same Bert-based semantic vectors as this paper. For the Weibo and PHEME datasets, we evaluated the accuracy (Acc.) of the two categories and the precision (Prec.), recall (Rec.), and F1 value (F1) of each category. The hidden feature vector dimension of each node was 64, and the dropout was set to 0.5. The training procedure was iterated for 100 epochs, and early stopping[28] was used when the validation loss stopped decreasing by 7 epochs.

C. Results and analysis

Table IV and Table V show the experimental results of the proposed method with all the comparison methods on the Weibo and PHEME datasets.

TABLE IV. RUMOR DETECTION RESULTS ON WEIBO DATASET

Method	Acc.	rumor			Non-rumor		
		Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
DTC	0.831	0.847	0.815	0.831	0.815	0.847	0.830
SVM-TS	0.857	0.839	0.885	0.861	0.878	0.830	0.857
DTR	0.732	0.738	0.715	0.726	0.726	0.749	0.737
GRU	0.899	0.865	0.946	0.904	0.940	0.852	0.894
RvNN	0.928	0.914	0.951	0.932	0.934	0.905	0.919
Bi-GCN	0.954	0.949	**0.956**	0.952	0.951	0.955	0.953
Re-GCN	**0.968**	**0.982**	0.954	**0.967**	**0.954**	**0.982**	**0.967**

TABLE V. RUMOR DETECTION RESULTS ON PHEME DATASET

Method	Acc.	rumor			Non-rumor		
		Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
DTC	0.670	0.572	0.435	0.494	0.687	0.837	0.755
SVM-TS	0.717	0.318	0.541	0.405	0.832	0.786	0.814
DTR	0.657	0.472	0.239	0.317	0.695	0.867	0.772
GRU	0.775	0.667	0.643	0.658	0.825	0.840	0.832
RvNN	0.820	0.733	0.741	0.731	0.869	0.857	0.867
Bi-GCN	0.828	**0.771**	0.714	0.736	0.858	**0.887**	0.871
Re-GCN	**0.845**	0.767	**0.781**	**0.772**	**0.886**	0.879	**0.882**

First, it can be clearly observed that the deep learning-based methods (GRU, RvNN, Bi-GCN, and Re-GCN) significantly outperform the manual feature-based methods (DTC, SVM-TS, and DTR). This is mainly due to the ability of deep learning methods to learn advanced representations of rumors and thus capture more effective features. This proves the importance of studying deep learning methods for rumor detection.

Secondly, among the deep learning-based methods, RvNN, Bi-GCN, and Re-GCN perform better relative to GRU since they consider the rumor propagation structure, while GRU ignores important rumor propagation structure features. It can also be observed that RvNN performs poorly relative to Bi-GCN due to the fact that RvNN is tree-structured by nature and it uses the hidden features of all leaf nodes as the final event representation, so it is more susceptible to the latest postings, resulting in the loss of more information from previous posts.

Finally, our proposed Re-GCN method is significantly better than the Bi-GCN method. There are two main reasons. First, Bi-GCN only considers the static propagation structure,

and Re-GCN takes into account the important factor of the dynamically changing propagation structure of rumors. Second, compared with Bi-GCN, which only uses textual content to represent events, Re-GCN considers both textual content and user information. This demonstrates the effectiveness of introducing user information and dynamically changing the propagation structure for rumor detection.

D. Ablation study

In order to analyze the effect of introducing user information and dividing the number of stages N, we compared different values of N and the use of text-only content with the introduction of user information.

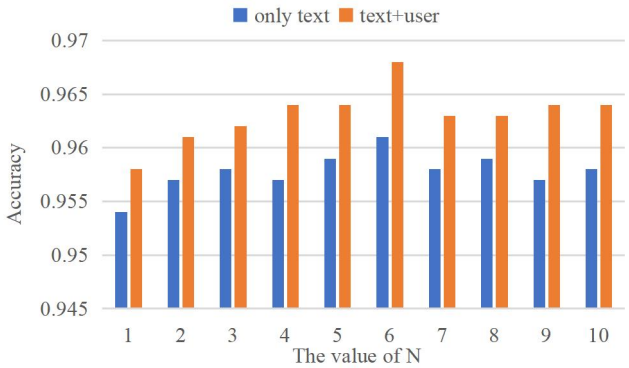


Figure 5. Experimental results for different values of N.

The experimental results on the Weibo dataset are shown in Fig.5, where the range of N is set to 1-10, "only text" means using only text as the post content representation, and "text+user" means splicing text and user information as the post content representation. According to Fig.5, firstly, it is evident that the accuracy rate of both datasets increases to different degrees after the introduction of user information, which indicates the effectiveness of introducing user information. Second, starting from a value of N of 1, the accuracy of rumor detection increases gradually as N's value increases, reaching the best result when N is taken as 6. When N is greater than 6, the model's accuracy decreases slightly but still outperforms all compared benchmark models, which indicates the effectiveness of the propagation structure considering dynamic changes.

In order to investigate the effect of the number of graph convolution layers on the model detection, we conducted experiments with the number of graph convolution layers set to 1, 2, 4 and 6, and the experimental results on the Weibo dataset are shown in Fig.6. According to Fig.6, it is not the case that the larger the number of graph convolution layers is, the better the overall performance is when the number of graph convolution layers is set to 2. Since increasing the depth of the graph convolution layers brings more parameters to the model, and the amount of data used for training is relatively insufficient, resulting in a possible overfitting of the model, the detection accuracy no longer improves with the increase in the number of graph convolution layers.

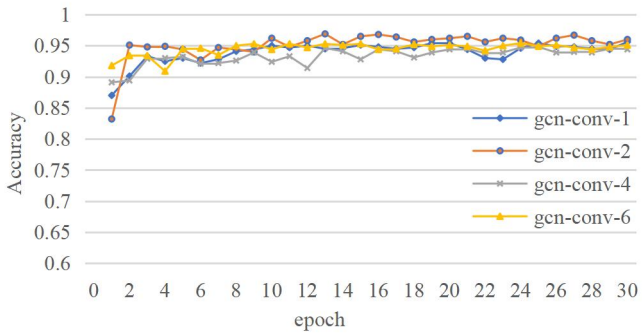


Figure 6. Experimental results of different graph convolution layers.

E. Early rumor detection

Early rumor detection is one of the important metrics for evaluating rumor detection methods, aiming to assess the ability of the method to detect rumors at an early stage of dissemination. We set a series of cutoff times and use only posts published before the cutoff time to compare the proposed method with the benchmark method.

Fig.7 shows the performance of our Re-GCN method and some benchmark methods for the Weibo dataset with different cutoff times. As can be seen from the Fig.7 Re-GCN achieves high rumor detection accuracy early in the propagation and outperforms the compared benchmark method at different cutoff times. This shows that considering user characteristics and dynamic propagation structure not only facilitates long-term rumor detection, but also helps to detect rumors early in propagation.

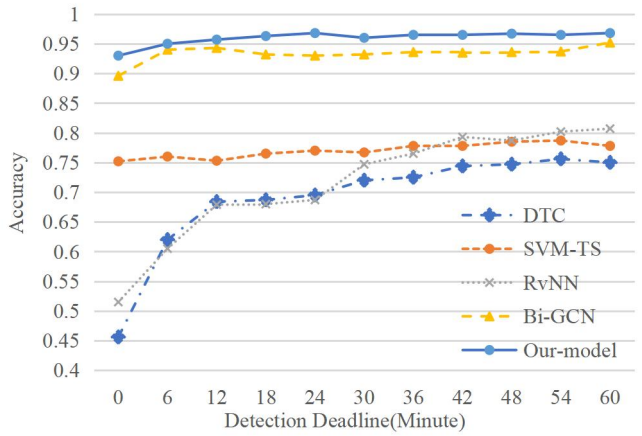


Figure 7. Experimental results of early rumor detection.

F. Case study

Take as an example the event in the PHEME dataset whose source tweet id is 500235112785924096, the propagation structure and some text contents of the posts in this event are shown in Fig.8. The event is labeled as a true rumor, which is incorrectly classified as a non-rumor by Bi-GCN using a static propagation structure, and correctly classified as a true rumor by our Re-GCN.

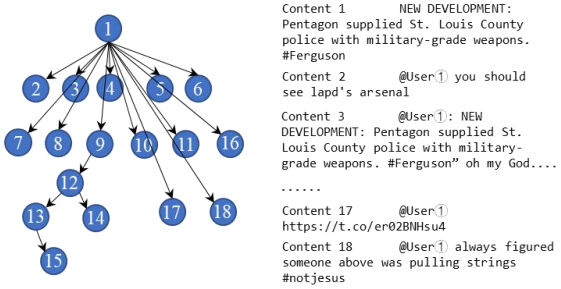


Figure 8. Propagation structure of the event with source twitter id 500235112785924096.

We extracted the attention values of the 6 stages when Re-GCN classified the event, as shown in Table VI. It can be seen that Re-GCN correctly classifies the event with the main clues of 3rd and 5th stage of the propagation of this event. As described in subsection *D* of Section IV, our proposed model is able to focus on diffierent stage of propagation features with the attention mechanism, which is beneficial to improve the effectiveness of rumor detection.

TABLE VI. ATTENTION VALUES IN 6 STAGES OF EVENT PROPAGATION

Terms	**Stage**					
	1st	***2nd***	***3rd***	***4th***	***5th***	***6th***
Attention Values	0	0	24.1	0	75.7	0.15

VI. CONCLUSION

This paper proposes a dynamic propagation structure-based rumor detection model named Re-GCN. First, compared with previous studies that mainly consider textual content, we encode both user information and textual content as post content representation, which provides richer information for the rumor detection model. Second, we introduced a dynamic propagation structure and constructed a dynamically changing propagation structure representation by dividing rumor propagation into multiple stages. Finally, the dynamically changing propagation features are captured using Bi-RNN and GCN, and an attention mechanism fuses the dynamic propagation features. Experimental results on two real-world datasets show that the proposed approach outperforms other state-of-the-art models for the rumor detection task.

REFERENCES

[1] Weibo:The 2020 Annual Report On counter-Rumor Work. https://weibo.com/1866405545/K0QaImwsK

[2] Castillo C, Mendoza M, Poblete B. Information credibility on twitter[C]//Proceedings of the 20th international conference on World wide web. 2011: 675-684.

[3] Popat K. Assessing the credibility of claims on the web[C]//Proceedings of the 26th International Conference on World Wide Web Companion. 2017: 735-739.

[4] Sampson J, Morstatter F, Wu L, et al. Leveraging the implicit structure within social media for emergent rumor detection[C]//Proceedings of the 25th ACM international on conference on information and knowledge management. 2016: 2377-2382.

[5] Ma J, Gao W, Wong K F. Detect rumors in microblog posts using propagation structure via kernel learning[C]. Association for Computational Linguistics, 2017.

[6] Yang F, Liu Y, Yu X, et al. Automatic detection of rumor on sina weibo[C]//Proceedings of the ACM SIGKDD workshop on mining data semantics. 2012: 1-7.

[7] Wu K, Yang S, Zhu K Q. False rumors detection on sina weibo by propagation structures[C]//2015 IEEE 31st international conference on data engineering. IEEE, 2015: 651-662.

[8] Liu X, Nourbakhsh A, Li Q, et al. Real-time rumor debunking on twitter[C]//Proceedings of the 24th ACM international on conference on information and knowledge management. 2015: 1867-1870.

[9] Yu F, Liu Q, Wu S, et al. A Convolutional Approach for Misinformation Identification[C]//IJCAI. 2017: 3901-3907.

[10] Ma J, Gao W, Mitra P, et al. Detecting rumors from microblogs with recurrent neural networks[J]. 2016.

[11] Ma J, Gao W, Wong K F. Rumor detection on twitter with tree-structured recursive neural networks[C]. Association for Computational Linguistics, 2018.

[12] Bian T, Xiao X, Xu T, et al. Rumor detection on social media with bi-directional graph convolutional networks[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(01): 549-556.

[13] Wei L, Hu D, Zhou W, et al. Towards Propagation Uncertainty: Edge-enhanced Bayesian Graph Convolutional Networks for Rumor Detection[J]. arXiv preprint arXiv:2107.11934, 2021.

[14] Lao A, Shi C, Yang Y. Rumor detection with field of linear and non-linear propagation[C]//Proceedings of the Web Conference 2021. 2021: 3178-3187.

[15] He Z, Li C, Zhou F, et al. Rumor Detection on Social Media with Event Augmentations[C]//Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021: 2020-2024.

[16] Kwon S, Cha M, Jung K, et al. Prominent features of rumor propagation in online social media[C]//2013 IEEE 13th international conference on data mining. IEEE, 2013: 1103-1108.

[17] Zhang Q, Zhang S, Dong J, et al. Automatic detection of rumor on social network[M]//Natural Language Processing and Chinese Computing. Springer, Cham, 2015: 113-122.

[18] Chen T, Li X, Yin H, et al. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection[C]//Pacific-Asia conference on knowledge discovery and data mining. Springer, Cham, 2018: 40-52.

[19] Zubiaga A, Liakata M, Procter R. Learning reporting dynamics during breaking news for rumour detection in social media[J]. arXiv preprint arXiv:1610.07363, 2016.

[20] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[21] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[22] Fang W, Lu L. Deep Graph Attention Neural Network for Click-Through Rate Prediction[C]//SEKE. 2020: 483-488.

[23] Liu X, You X, Zhang X, et al. Tensor graph convolutional networks for text classification[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(05): 8409-8416.

[24] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.

[25] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[26] Ma J, Gao W, Wei Z, et al. Detect rumors using time series of social context information on microblogging websites[C]//Proceedings of the 24th ACM international on conference on information and knowledge management. 2015: 1751-1754.

[27] Zhao Z, Resnick P, Mei Q. Enquiring minds: Early detection of rumors in social media from enquiry posts[C]//Proceedings of the 24th international conference on world wide web. 2015: 1395-1405.

[28] Yao Y, Rosasco L, Caponnetto A. On early stopping in gradient descent learning[J]. Constructive Approximation, 2007, 26(2): 289-315.

Chinese Spam Detection based on Prompt Tuning

Yan Zhang
College of Computer Science
Inner Mongolia University
Hohhot, China
zy09230@163.com

Chunyan An*
College of Computer Science
Inner Mongolia University
Hohhot, China
ann@imu.edu.cnd

***Abstract*—Spam has plagued Internet users for a long time, and it is of great significance to design an efficient spam detection method. In recent years, spam detection methods based on fine-tuning pre-trained language models (PLM) have achieved great success. The approach is to fine-tune a pre-trained language model on a large dataset to adapt it to the downstream spam detection task. However, the objective of the initial training phase of PLM is inconsistent with the objective of downstream tasks, which results in the downstream tasks cannot fully utilize the latent knowledge in PLM. In this paper, we use Prompt Tuning and PLM to identify Chinese spam by constructing additional prompt templates, converting the email classification task into a fill-in-the-blank task, and then getting the email classification results according to the filling content on the prompt templates. This process is very similar to the process of initial training of PLM, which can more fully utilize the rich knowledge in PLM. We use prompt tuning to train the model on public datasets. Through experiments, we found that the accuracy score of the proposed model on trec06 datasets can reach 0.996, and the F1 score can reach 0.994, which is better than the comparison model. In terms of model convergence speed, the proposed model only needs less than 200 training steps to converge, which is faster than the comparison model.**

Keywords-Chinese spam detection; deep learning; prompt tuning; ERNIE

I. INTRODUCTION

The number of Internet users in China has grown rapidly in the past 10 years due to the booming of traditional Internet and mobile Internet. According to "The 44th China Statistical Report on Internet Development", as of June 2019, China's number of Internet users has reached 854 million [1]. This rapid development also facilitates the dissemination of information, including social media, email, WeChat platforms, and other applications. At the same time, these channels also attract malicious users to spread spam, threatening people's property safety. The goal of spam detection technology is to filter out spam before it causes damage to users.

Current mainstream spam detection methods are based on machine learning (ML) and deep learning (DL) techniques. The characteristics of emails are learned through sample data to classify emails. Models commonly used for spam detection include support vector machines (SVM) [2], convolutional neural networks (CNN) [3], recurrent neural networks (RNN) [4], etc. How to obtain accurate text features has a huge impact on model performance, and the pre-trained language models (PLM) that have appeared in recent years have effectively solved this problem. PLM has achieved good results in multiple NLP tasks, and the effect of email detection models can be significantly improved by PLM [5]. The approach is to fine-tune PLM through the task target dataset to make it suitable for downstream tasks. However, an important problem is that the target task of the initial training of PLM is the filling-in-the-blank task, and the downstream task is the classification task, which will cause the model to fail to fully utilize the knowledge in the PLM.

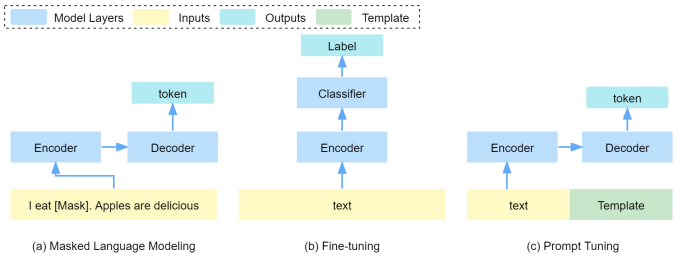


Figure 1. Pre-trained language model training process (a), and two paradigms of fine-tuning (b) and prompt tuning (c).

Now, a new paradigm called prompt tuning [6] has achieved satisfactory results in tasks such as news classification. It does not directly classify through the features of the text but designs prompt templates to convert downstream tasks into similar to the initial training of PLM, let PLM directly complete the task of filling in the blanks, making more efficient use of the rich knowledge in PLM, as shown in Figure 1. However, spam classification is somewhat different from news classification: in order to evade detection, spam will use various methods to disguise as normal emails, and even express friendly feelings, while normal emails may contain some negative or even offensive Emotion, which brings greater challenges for PLM to fill in the blanks in the prompt templates.

The contributions of this paper are summarized as follows: We design a Chinese spam detection model based on prompt tuning and ERNIE [7]. We construct prompt templates as additional information to help the ERNIE model achieve spam detection. We conduct experiments on the public Chinese spam dataset, and the experiments show that our model outperforms existing models in terms of accuracy score, F1 score, and convergence speed.

*Corresponding Author
DOI reference number: 10.18293/SEKE2022-120

II. Related Work

Over the years, many methods have been proposed to detect spam based on the content of the email. These methods fall into three main categories: supervised, semi-supervised[8], and unsupervised[9] methods. Supervised methods treat mail detection as a classification task, and supervised methods generally show better performance compared to other methods[10].

Among the various methods of supervised learning, methods based on deep learning perform better. [11] tested many supervised learning algorithms such as NB, SVM, KNN, etc., which were used individually or in combination to detect spam. [12] found that ML methods are inefficient in the case of high-dimensional data and different spammers, and it becomes crucial to explore DL methods with effective feature selection mechanisms. With the development of deep learning, many scholars began to use DL technology to automatically learn features. In NLP, deep learning methods are mainly based on a distributed representation of each word, also known as word embedding [13]. [14] implemented a Semantic Convolutional Neural Network (SCNN) model that maps word vectors using an NLP technique called Word2Vec. [5] shows that different methods of obtaining word vectors have a greater impact on feature extraction, and using PLM to obtain word vectors is the best way.

The common approach in existing works is to extract features using encoders of pre-trained language models and then use classification algorithms to identify spam. Spam will deceive the detection model by carefully designing the input content, which requires the detection model to obtain sufficiently accurate email features. [5] used BERT Encoder to obtain email features and tested a variety of classification algorithms. In addition, they simulated the camouflage methods commonly used in spam such as synonym replacement to test the performance of the model. The results show that the BERT-based detection model can effectively resist this camouflage strategy. [15] used the M-BERT pre-trained language model, which can encode multiple languages to achieve multilingual mixed spam classification.

The spam detection model designed in this paper is aimed at Chinese spam, and we select the ERNIE model to extract the text features of the email. The ERNIE model has achieved excellent results in some Chinese natural language processing tasks and demonstrated strong knowledge reasoning ability in the cloze test. ERNIE uses multi-layer Transformer [16] as basic encoder. The Transformer can capture the contextual information for each token in the sentence via self-attention, and generates a sequence of contextual embeddings. In the training stage, they randomly mask 15 percents of basic language units, and using other basic units in the sentence as inputs, and train a transformer to predict the mask units [7].

In past studies, the standard paradigm for using pre-trained language models to handle downstream tasks is the fine-tuning paradigm [17], which focuses on designing training objectives in the training phase to adapt pre-trained language models to downstream tasks. Today, this paradigm is hopefully replaced by a paradigm called prompt tuning, which reformulates downstream tasks with the help of textual cues to look more like the tasks solved during the original PLM training [6]. Some studies have applied this paradigm in text classification tasks, such as [18] formulating task-specific prompt templates according to logical rules, outperforming baseline models in many-class text classification. [19] showed that training models with prompt templates on different tasks or different amounts of data can improve the prediction performance.

Although the use of the prompt-tuning paradigm in some classification tasks can improve the model effect, since spam emails are disguised as normal emails using various methods, there is no relevant research to show that this paradigm is still suitable for spam detection models. In this paper, a spam detection model based on the ERNIE and prompt tuning is designed for Chinese spam. The prompt template is manually set, and PLM completes the task of filling in the template. Then the model classifies the email according to the content filled in the template by PLM.

III. Prompt Tuning Method

An example of email detection with prompt tuning is shown in Figure 2. Our model consists of the following parts:

- Prompt Addition: Design a suitable prompt template for the task. The content of the template is a sentence describing the email message, and it has a [Mask] placeholder. This template is spliced with the original text and used as the input of the Encoder.
- Fill in the Prompt template: The input text is encoded and decoded by PLM, and the filling content of the [Mask] position is obtained.
- Answer Mapping: Mapping the filled content Y of the [Mask] position with the answer space Z, resulting in an effective predictive model.

A. Prompt Addition

A classification task can be denoted as $T = \{X, Y\}$, where X is the instance set and Y is the class set. For each $x \in X$, there is a unique label $y \in Y$.

Prompt Addition needs to design a prompt template T() suitable for the task first, and then map each text x to $x_{prompt} = T(x)$, x_{prompt} is obtained by splicing the prompt template and the email text. The prompt template converts the original task into a fill-in-the-blank task by adding additional prompt information and at least one [Mask] placeholder. We set T() = 'This is a [Mask][Mask] email' and x maps to x_{prompt} = 'This is a [Mask][Mask] mail. x '. In addition, we set a character set V, $v \in V$ to fill the prompt template.

B. Fill in the Prompt template

Calculate the conditional probability $p([MASK] = v| x_{prompt})$ of the [Mask] position padding character using the encoder-decoder network structure from PLM. First add the token embedding, position embedding, and segment embedding of x_{prompt} as the input of the encoder.

Encoder uses multi-layer Transformer structure to capture the context of each character through self-attention to encode the

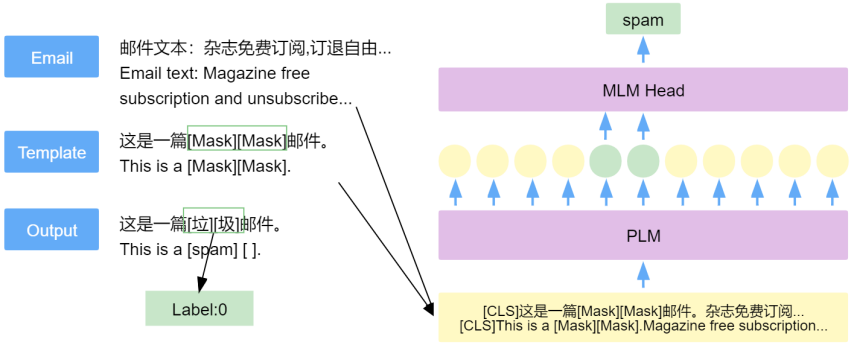


Figure 2. An example of implementing email detection using prompt tuning.

input content. The Multi-Head Attention in the Decoder decodes according to the output from the Encoder, and outputs the hidden vector $h_{[Mask]}$ of [Mask]. Calculate the dot product of $h_{[Mask]}$ and Emb(v), Emb(v) is the embedding of the token v. Then, the scores of all characters in the model dictionary are calculated by the softmax function, and the character with the largest score is obtained as the predicted character. To prevent the model from predicting characters that exceed our answer space, we restrict the output to the set V.

$$p([Mask] = v|x_{prompt}) = \begin{cases} \text{Softmax}\left(h_{[Mask]} \cdot \text{Emb}(v)\right) & v \in V \\ 0 & v \notin V \end{cases} \quad (1)$$

C. Answer Mapping

At this stage, the calculated predicted characters need to be mapped to the category labels of the emails. Set a mapping $\emptyset$ between the mail category y and the predicted character v. With this mapping function, we can map the predicted category label to the predicted character at the [Mask] position:

$$p(y|x) = p([Mask] = \emptyset(y)|x_{prompt}) \quad (2)$$

For example, we can map the label of normal mail to 'good' and the label of spam to 'bad'. When the pre-trained language model fills in 'good' or 'bad' on the prompt template, we can know whether the email x is spam.

Finally, according to the dataset D, the cross-entropy loss function is used to calculate the error between the correct answer and the predicted value, and all the model parameters are updated.

$$Loss = -\frac{1}{|X|}\sum_{x \in X} \log p([Mask] = \emptyset(y_x)|T(x)) \quad (3)$$

IV. EXPERIMENTS

We conduct experiments on Chinese datasets to demonstrate the effectiveness of our model in the task of email detection.

A. Datasets and Experimental Settings

We conduct experiments on public Chinese spam datasets: Trec06[20] and microblogPCU[21].

- Trec06： one of the largest and most widely used datasets for Spam Detection， including Chinese and English emails.

TABLE I. STATISTICS OF DIFFERENT DATASETS

Dataset	*SPAM*	*HAM*	*Total*
Trec06	42853	21766	64619
microblogPCU	2194	3007	5201

- microblogPCU: Data sourced from Weibo, including email text and information about these spammers.

More details of these datasets are shown in Table I shows. For all the above datasets, we use F1 scores and accuracy scores as the main metric for evaluation.

B. Experimental settings

We set the length of each text to 250 characters, excess characters are deleted, and insufficient characters are padded with 0 when converted into word embeddings. our model is optimized with Adam using the learning rate of 1e - 5 on ERNIE-1.0, with a linear warmup for the first 10% steps. For all datasets, we train our model for 20 epochs with the batch size 64. The best model checkpoint is selected based on the performance on the development set.

C. baseline models

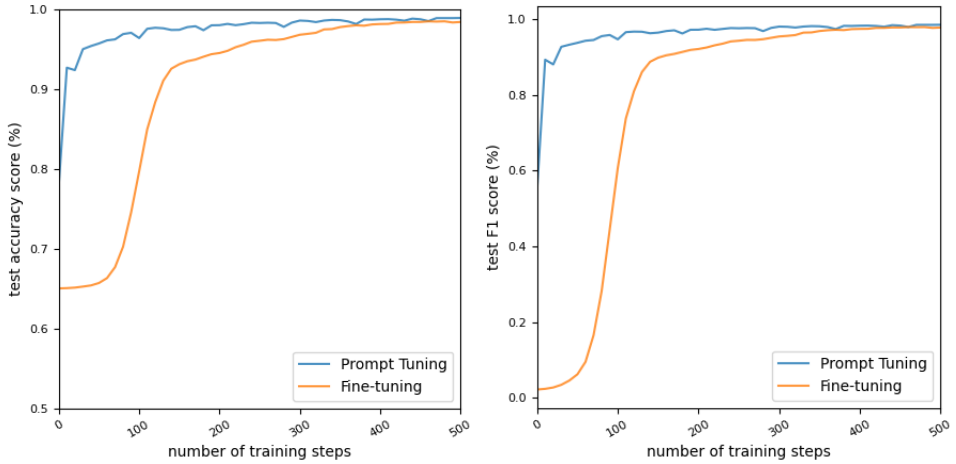
we compare our model with several typical models for text classification, including:

- Learning models from scratch: for text classification, the typical approach is learning neural models from scratch. We choose SVM [2] , Naive Bayes ,DPCNN [3], TextRNN [4], TextRCNN [22] as baselines, these models perform well in text classification tasks.
- Fine-tuning pre-trained models: PLM performs well in various NLP tasks, and many works use fine-tuning PLM for text classification. [15] showed that using PLMs is effective for spam detection, we implemented fine-tuning on ERNIE.

In addition, due to the strong semantic expression ability of PLM, we improved the deep learning techniques mentioned above by using PLM as an embedding layer, implementing ERNIE+RCNN, ERNIE+TextRNN, and ERNIE+DPCNN.

TABLE II. COMPARATIVE EXPERIMENTAL RESULTS OF OUR MODEL AND BASELINE MODELS ON DIFFERENT DATASETS.

	Models	Trec06		microblogPCU	
		Accuracy	*F1*	*Accuracy*	*F1*
Learning models from scratch	Naive Bayes	0.9758	0.9734	0.7958	0.7874
	SVM	0.9935	0.9929	0.7861	0.7834
	TextRNN	0.9882	0.9833	0.7418	0.7994
	TextRCNN	0.9943	0.9915	0.8170	0.8393
	DPCNN	0.9948	0.9926	0.8035	0.8277
Fine-tuning pre-trained models	ERNIE+TextRNN	0.9950	0.9928	0.8265	0.8495
	ERNIE+ TextRCNN	0.9953	0.9933	0.8170	0.8398
	ERNIE+ DPCNN	0.9931	0.9900	0.7765	0.8159
	Fine-tuning ERNIE	0.9951	0.9931	0.8324	0.8533
Prompt tuning pre-trained models	prompt tuning ERNIE	**0.9960**	**0.9942**	**0.8423**	**0.8633**



(a) Accuracy scores (b) F1 scores

Figure 3. Changes in accuracy scores (a) and F1 scores (b) with increasing number of training steps.

D. *Experimental results*

The experimental results are shown in Table II, From the table, we can see that:

- Using a pre-trained language model to obtain text features can significantly improve the model performance compared to methods that do not use a pre-trained model.
- For the Fine-tuning paradigm, using a pre-trained language model combined with DPCNN, TextRNN and TextRCNN have a small performance gap compared to just using a pre-trained language model and a fully connected layer for classification.
- Our model uses the prompt tuning paradigm, and its performance is better than the model using the fine-tuning paradigm. The reason for the low F1 scores and accuracy scores on the microblogPCU dataset is that the length of the text is short, and the dataset itself has problems with labeling errors.

In general, the experimental results in Table II show that our model has higher F1 scores and accuracy scores than the baseline models in the Chinese spam detection task. We believe that this is the result of the combined knowledge of the prompt template and the latent knowledge of the PLM.

We also tested the effect of training sets of different sizes on the model, as shown in Figure 4, our model performed well on both small and rich datasets. As shown in Figure 3, training the model through prompt tuning can significantly improve the convergence speed. Our model only requires less than 200 training steps to converge, while training the model with fine-tuning requires 400 training steps. Furthermore, our model had an F1 score of 0.6 before the model started training, while the comparison model only had an F1 score of 0.2. Our experiments show that the knowledge contained in the prompt template plays a positive role in both model convergence and prediction.

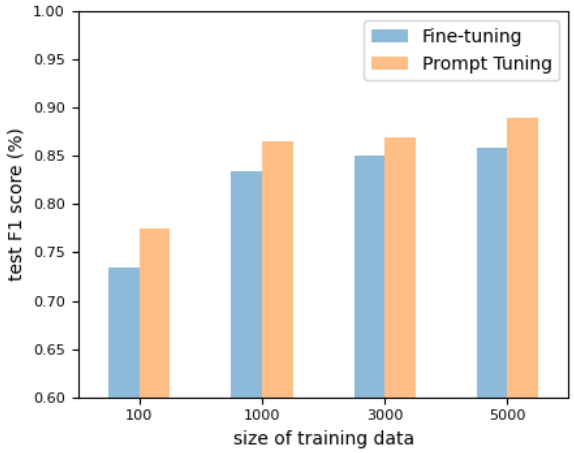


Figure 4. Comparison of F1 scores (%) when training models on different sized training sets with two paradigms.

V. CONCLUSION

In this paper, we design a Chinese spam detection model using the ERNIE pre-trained language model and the prompt tuning paradigm. By designing the prompt template, the mail classification task is converted into a filling-in-the-blank task, and the latent knowledge of the pre-trained language model and the knowledge contained in the prompt template are more fully utilized, which improves the convergence speed and prediction accuracy of the model. The experimental results show that the accuracy score of our model can reach 0.996, the F1 score can reach 0.994, which is better than the comparison model, and the convergence speed of our model is faster. In our work, the design of prompt template relies on the expertise of developers. In the future, we hope to further study the construction method of prompt templates, try to automatically generate templates and design learnable templates.

VI. ACKNOWLEDGEMENT

The authors wish to thank the Project of Inner Mongolia Science &Technology Plan under Grant No. 2021GG0164, 2020GG0186, Natural Science Foundation of China under Grant No.61862047,61962039,62162046, Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering, Inner Mongolia Engineering Lab of Cloud Computing and Service Software and Inner Mongolia Engineering Lab of Big Data Analysis Technology.

REFERENCES

[1] http://www.cac.gov.cn/2019/08/30/c_1124938750. htm.

[2] S. I. Wang and C. D. Manning, “Baselines and Bigrams: Simple, Good Sentiment and Topic Classification,” in The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers, 2012, pp. 90–94, [Online]. Available: https://aclanthology.org/P12-2018/.

[3] R. Johnson and T. Zhang, “Deep Pyramid Convolutional Neural Networks for Text Categorization,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, 2017, pp. 562–570, doi: 10.18653/v1/P17-1052.

[4] P. Liu, X. Qiu, and X. Huang, “Recurrent Neural Network for Text Classification with Multi-Task Learning,” in Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, 2016, pp. 2873–2879.

[5] S. R. Galeano, “Using BERT Encoding to Tackle the Mad-lib Attack in Spam Detection,” CoRR, vol. abs/2107.0, 2021.

[6] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing,” CoRR, vol. abs/2107.1, 2021, [Online]. Available: https://arxiv.org/abs/2107.13586.

[7] Y. Sun et al., “ERNIE: Enhanced Representation through Knowledge Integration,” CoRR, vol. abs/1904.0, 2019, [Online]. Available: http://arxiv.org/abs/1904.09223.

[8] J. S. Whissell and C. L. A. Clarke, “Clustering for semi-supervised spam filtering,” in The 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS 2011, Perth, Australia, September 1-2, 2011, Proceedings, 2011, pp. 125–134, doi: 10.1145/2030376.2030391.

[9] Z. Chen and D. Subramanian, “An Unsupervised Approach to Detect Spam Campaigns that Use Botnets on Twitter,” CoRR, vol. abs/1804.0, 2018, [Online]. Available: http://arxiv.org/abs/1804.05232.

[10] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, “Machine learning for email spam filtering: review, approaches and open research problems,” Heliyon, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.

[11] R. Narayan, J. K. Rout, and S. K. Jena, “Review Spam Detection Using Opinion Mining,” in Progress in Intelligent Computing Techniques: Theory, Practice, and Applications, 2018, pp. 273–279.

[12] A. Barushka and P. Hajek, “Spam Filtering in Social Networks Using Regularized Deep Neural Networks with Ensemble Learning,” in Artificial Intelligence Applications and Innovations, 2018, pp. 38–49.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” in 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, 2013, [Online]. Available: http://arxiv.org/abs/1301.3781.

[14] G. Jain, M. Sharma, and B. Agarwal, “Spam Detection on Social Media Using Semantic Convolutional Neural Network,” Int. J. Knowl. Discov. Bioinform, vol. 8, no. 1, pp. 12–26, 2018.

[15] J. Cao and C. Lai, “A Bilingual Multi-type Spam Detection Model Based on M-BERT,” in IEEE Global Communications Conference, GLOBECOM 2020, Virtual Event, Taiwan, December 7-11, 2020, pp. 1–6, doi: 10.1109/GLOBECOM42002.2020.9347970.

[16] [A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 5998–6008, [Online]. Available:https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[17] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained Models for Natural Language Processing: A Survey,” CoRR, vol. abs/2003.0, 2020, [Online]. Available: https://arxiv.org/abs/2003.08271.

[18] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, “PTR: Prompt Tuning with Rules for Text Classification,” 2021, [Online]. Available: http://arxiv.org/abs/2105.11259.

[19] T. Le Scao and A. M. Rush, “How many data points is a prompt worth?,” in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, [Online], 2021, pp. 2627–2636, doi: 10.18653/v1/2021.naacl-main.208.

[20] https://plg.uwaterloo.ca/˜gvcormac/treccorpus06/,2006,[Online].

[21] https://archive.ics.uci.edu/ml/datasets/microblogPCU,2015,[Online].

[22] S. Lai, L. Xu, K. Liu, and J. Zhao, “Recurrent Convolutional Neural Networks for Text Classification,” in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 2267–2273.

Identifying Gambling Websites with Co-training

Chenyang Wang ★ Pengfei Xue ★ Min Zhang ★ Miao Hu ★

★ National University of Defense Technology

Abstract

Gambling websites do great harm to society and many even cause serious network crime. To identify the gambling websites, many machine learning based methods are proposed by analysing the URL, the text, and the images of the websites. Nevertheless, most of them ignore one important information, i.e., the text within the website images. The text on the images of gambling websites has keywords that clearly point to such websites. Motivated by this, in this paper, we propose an co-training based gambling website identification method by combining the visual and semantic features of the website screenshots. First, we extract text information from webpage screenshots through the optical character recognition (OCR) technique. Then we train an image classifier based on a convolutional neural network (CNN) and a text classifier based on TextRNN respectively from image view and text view. Second, the two classifiers are retrained on unlabeled data with the co-training algorithm. Third, we conduct experiments on the webpage screenshot dataset we collected. The experimental results indicate that OCR text has strong semantic feature and the proposed method can effectively improve the performance in identifying gambling websites.

Index terms— *Co-training, Convolutional Neural Network, TextRNN.*

1 Introduction

Nowadays, most people get information from the Internet. However, the Internet is full of malicious content, especially gambling websites, which are on the edge of network crime and do great harm to the society. Due to the huge number and continuous updating of gambling websites, it is difficult to identify manually. Therefore, it is necessary to design an automatic, efficient, and accurate method to identify gambling websites.

The existing malicious website identification methods could be classified into black-list based, URL based [1, 2, 3], webpage content based [4, 5, 6, 7] and mixed-features based [8, 9]. Black-list based methods establish a black list by collecting the malicious URLs or domain names. It is labor-intensive to establish and maintain the black list, and the detection efficiency is slow. URL based methods extract features from URLs for classification. However, because the URLs contain insufficient information, the identification accuracy of URL based methods is not high. Webpage content based methods extract content features from webpages for identification, such as HTML text, image, link, JavaScript code, etc. Mixed-feature based methods combine different features to improve the accuracy of classification. For both webpage content based methods and mixed-feature based methods, when extracting visual features from images for gambling websites identification, the accuracy of them is not too high due to the high complexity of the image.

Figure 1: The text information on the website images

However, in our study, we find that there are some text information on website images that has strong semantic features and can be used to identify gambling websites. Figure 1 shows an example of a gambling website that has the text ”投注0%风险,稳赚不赔” (the text in English means that ”there is no risk of betting and you can earn money without loss”). The word ”投注” (”betting”) clearly points to gambling. In order to avoid the keyword matching detection methods, some gambling websites do not have such obvious keywords in the html text, but in the website images. There are two challenges to solve the problem: the one challenge is how to extract the text information within the images of websites. The other challenge is how to combine image and the text information to identify gambling websites.

Motivated by this, in this paper, we propose an co-training based gambling websites identification method, which extracts visual and semantic features of webpage screenshots, and utilizes unlabeled data to improve the performance of classification models. The idea of co-training is to benefit from two (or more) models which are trained from different views. These views may be obtained from multiple sources or different feature subsets (e.g., image is a view and text is another view). The two models are complementary to one another and can help correct each

DOI reference number: 10.18293/SEKE2022-106

other when they make mistakes. Naturally, the idea of co-training suits the task of learning a classification system from the image and the text view to identify gambling websites. Specifically, the proposed method is implemented as follows. Firstly, we extract text information from webpage screenshots by OCR [10] technique. Then, we train an image classifier based on CNN [11]. and a text classifier based on TextRNN [12] respectively from image view and text view. Finally, we retrain the image and text classifiers on unlabeled data with the co-training algorithm. The proposed method has the following advantages: (i) It extracts text information on website images which has strong semantic features and is useful in identifying gambling websites. (ii) It proposes a gambling websites identification method based on the multi-view semi-supervised learning algorithm (co-training) which uses unlabeled data to improve the performance of the classification model. (iii) It combines mixed-feature based identification method with semi-supervised learning to better identify gambling websites. The main contributions of this paper are as follows.

1. We propose an co-training based gambling websites identification method in this study. Compared to existing methods, this method make full use of the visual feature and key semantic feature of webpage screenshots. Through the multi-view semi-supervised learning (co-training) algorithm, this method utilizes a large number of unlabeled data to improve the performance of classification models.
2. We use OCR technique to extract the text information on webpage screenshots which has strong semantic features. Then, we train an image classifier based on CNN and a text classifier based on TextRNN respectively from image view and text view. Finally, we retrain the image and text classifiers on unlabeled data with the co-training algorithm to improve the performance of the two classifiers.
3. We evaluate the proposed method by conducting experiments in the gambling dataset we collected. The experimental results show that the proposed method can effectively improve the performance of classifiers in terms of precision, recall, and F1-score.

2 Background

Co-training. The traditional supervised learning method uses a large number of labeled data to establish a model and predict the labels of unknown instances. If only a small number of labeled data is used, the trained model is difficult to have strong generalization ability. Semi-supervised learning attempts to make the machine automatically use a large number of unlabeled data to assist a small number of labeled data in learning. The goal is to obtain the best generalization performance on these unlabeled data [13].

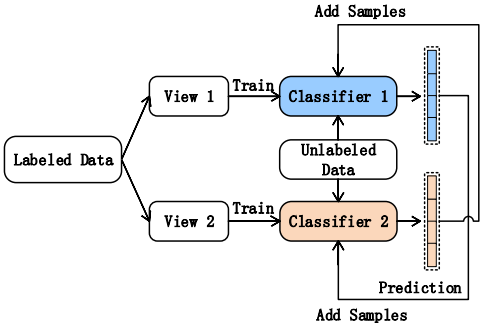


Figure 2: Base framework of co-training

Co-training [14, 15, 16] is a multi-view semi-supervised classification algorithm which improves the generalization performance of models by combining two (or more) classifiers trained from different views. Firstly, the labeled data is used to train the classifiers under different views. Then the classifiers predict on unlabeled data and label the samples with high prediction confidence. The samples that are labeled with pseudo-labels are added to the training datasets of other classifiers for retraining. Co-training is shown in Figure 2.

Different views exchange the prediction labels of unlabeled samples to realize the information exchange. The co-training algorithm is based on two key assumptions. The one assumption is that each view contains enough information to build the optimal learner. The other assumption is that the two (or more) views are independent under the condition of a given class label. Although the process of co-training algorithm is simple, the theory proves that if the two views satisfy the two key assumptions, the generalization of weak classifiers can be improved to any high level through co-training using unlabeled data. The two key assumptions are often difficult to satisfy in real tasks, so the performance improvement will not be so large. However, research shows that co-training can effectively improve the performance of weak classifiers [13]. Qiao et al. [17] present Deep Co-Training (DCT) based on co-training framework for semi-supervised image recognition, which improves the accuracy of models. Katz et al. [18] propose an ensemble-based co-training approach that make use of unlabeled text data to improve text classification when labeled data is very small.

3 Methodology

3.1 Overview of the Proposed Method

Figure 3 shows our method based on co-training for gambling website identification. The webpage screenshot data can be divided into two views: one is the image view data, and the other is the text view data on the image which can be extracted by OCR. Although the image and text data

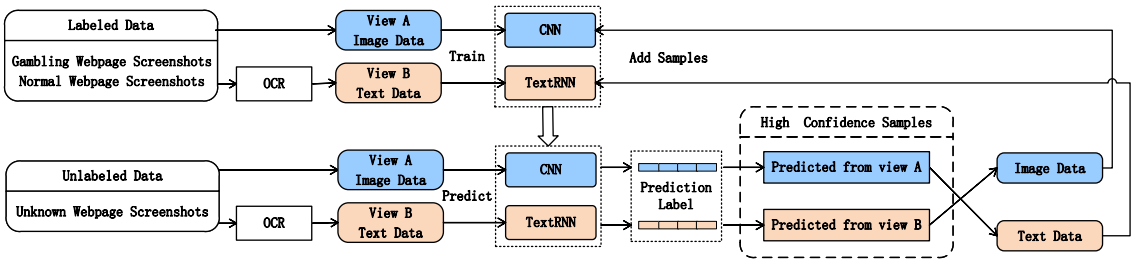


Figure 3: Overall framework of the proposed method.

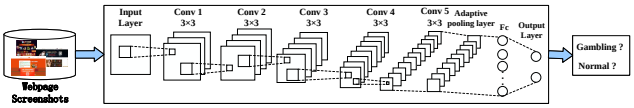


Figure 4: The image classifier based on CNN

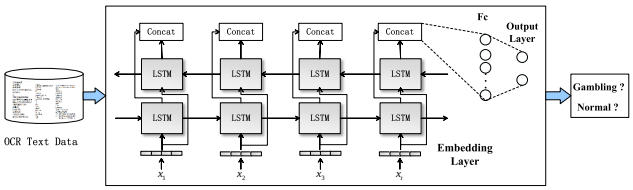


Figure 5: The text classifier based on TextRNN

are extracted from the same webpage screenshot, they are from different views, satisfying the sufficient redundancy and conditional independence of co-training.

First, we extract text messages on webpage screenshots by OCR technique. The webpage screenshots are the data of view A (image view) and the OCR texts are the data of view B (text view). We train an image classifier based on CNN and a text classifier based on TextRNN respectively from view A and view B on labeled data. Second, the unlabeled data are predicted from view A and view B. The samples with high prediction confidence are selected and labeled. The image data of the selected samples predicted from text view are added to the training dataset of the image classifier. Similarly, the text data of the selected samples predicted from image view are added to the training dataset of the text classifier. Third, the two classifiers are retrained with new training datasets.

3.2 Image Classifier based on CNN

The image classifier is constructed based on the CNN model. The input are webpage screenshots and the output of the fully-connected network are prediction labels, which are gambling or normal. The image classifier based on CNN is shown in Figure 4.

We stack 3×3 small convolutional kernels to construct convolutional networks instead of using large convolutional kernels like 7×7. Because the receptive field of a small convolutional kernel is smaller, it can extract finer-grained image features. In addition, a larger convolutional kernel requires more computation than a small one. Thus, we stack several 3×3 kernels to construct the CNN model. We add an adaptive pooling layer after convolutional layers so that the model can process the input of different sizes.

3.3 Text Classifier based on TextRNN

A text classifier is constructed based on TextRNN, the input is text data of webpage screenshots extracted by OCR technique. We use pre-trained word vectors as embedding representation, the output of hidden layer at moment t is semantic feature and input to fully-connected layer for classification. The text classifier based on TextRNN is shown in Figure 5.

We use the Sogou News Chinese corpus to train the pre-trained word vector as the embedding layer. Words in the word list are represented as word vectors, words not in the word list are represented as random vectors, and the dimension of word vectors is 300. In hidden layers we use Bi-LSTM, and each LSTM unit has the following vectors in the time step t: An input gate i_t, an output gate o_t, a forgetting gate f_t, a memory unit c_t, and hidden layer state h_t. The input gate i_t process the input of the current sequence position. The forgetting gate f_t determines what information is discarded from the cell.

3.4 Co-training of Image and Text Classifiers

First, we train image and text classifiers on labeled dataset $L = \{(x_i^A, x_i^B, y_i)|i = 1, \ldots, N\}$. Where x_i^A is the image data of sample x_i, x_i^B is the text data of sample x_i, y_i is the label, and N is the total number of samples. Second, the image classifier predicts on unlabeled dataset $U = \{(x_i^A, x_i^B)|i = 1, \ldots, M\}$, select p positive samples and q negative samples with high prediction confidence. Text data $\{(x_t^B, y_t)|t = 1, \ldots, p+q\}$ of selected samples is added to the training dataset L_B of the text classifiers. The labels $\{y_t|t = 1, \ldots, p+q\}$ are pseudo-labels. Similarly, $p + q$ samples with high prediction confidence

Algorithm 1 Co-training of image and text classifiers

Input:

Labeled dataset $L = \{(x_i^A, x_i^B, y_i)|i = 1, \ldots, N\}$,
Unlabeled dataset $U = \{(x_i^A, x_i^B)|i = 1, \ldots, M\}$,
Iteration number g,
The confidence threshold θ.

Output:

The trained image and text classifiers.

Process:

1: Train an image classifier based on CNN on labeled dataset $L_A = \{(x_i^A, y_i)|i = 1, \ldots, N\}$.
2: Train a text classifier based on TextRNN on labeled dataset $L_B = \{(x_i^B, y_i)|i = 1, \ldots, N\}$.
3: The image classifier predicts on unlabeled dataset U and selects p positive samples and q negative samples with confidence higher than θ. And these $p + q$ selected samples are added to the training dataset of the text classifier. The new dataset $L_B = \{(x_i^B, y_i)|i = 1, \ldots, N + p + q\}$.
4: The text classifier predicts on unlabeled dataset U and selects p positive samples and q negative samples with confidence higher than θ. And these $p + q$ selected samples are added to the training dataset of the image classifier. The new dataset $L_A = \{(x_i^A, y_i)|i = 1, \ldots, N + p + q\}$.
5: Remove the selected $2p + 2q$ samples from U.
6: Retrain the image classifier on the new dataset L_A.
7: Retrain the text classifier on the new dataset L_B.
8: Repeat steps 3 to 7 g times.

that predicted by the text classifier are added to the training dataset L_A of the image classifier. Third, image and text classifiers are retrained on new training datasets L_A and L_B. The above process is repeated several times, The algorithm is as Algorithm 1.

4 Experiments and Analysis

Datasets. We use crawlers to get webpage screenshots of the website on the Internet. We collect 1600 webpage screenshots, including 800 of gambling and 800 of normal (including movie, science, education, traffic, shopping, medical, etc.). These images are labeled to form the dataset L_A. We extract text data from webpage screenshots by OCR. These text data are preprocessed to form the dataset L_B. In addition, a lot of webpage screenshots are collected and used for unlabeled dataset U. A total of 600 webpage screenshots of gambling and normal websites constitute the test dataset T_A and 600 OCR text data constitute the test dataset T_B.

Evaluation metrics. In this paper, we use three evaluation metrics to evaluate the method, including Precision, Recall, and F1-score.

Table 1: Results of CNN-5 and Resnet34

Evaluation metrics	Model	Gambling	Normal	Overall
Precision	Resnet34	0.7639	0.8958	0.8299
	CNN-5	**0.8364**	**0.9111**	**0.8737**
Recall	Resnet34	0.9167	0.7167	0.8167
	CNN-5	**0.9200**	**0.8200**	**0.8700**
F1	Resnet34	0.8333	0.7963	0.8148
	CNN-5	**0.8762**	**0.8632**	**0.8697**

Table 2: Results of TextRNN on OCR text and html text

Evaluation metrics	Data	Gambling	Normal	Overall
Precision	html text	0.9052	0.9218	0.9135
	OCR text	**0.9794**	**0.9515**	**0.9654**
Rrcall	html text	0.9233	0.9033	0.9133
	OCR text	**0.9500**	**0.9800**	**0.9650**
F1	html text	0.9142	0.9125	0.9133
	OCR text	**0.9645**	**0.9655**	**0.9650**

4.1 Image Classification based on CNN

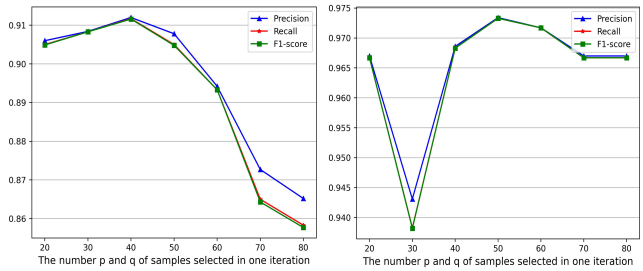
We construct an image classifier named CNN-5 and stack five 3×3 small kernels in convolutional layers as the feature extraction module. After the last convolutional layer, an adaptive pooling layer is added to unify the image output size, and then a fully-connected layer is connected as the classification module. We also do the same experiment on the pre-trained model Resnet34 for comparison. When using the Resnet34 pre-trained model, we freeze all feature extraction modules, that is, all convolution layers, and update the weight of the fully-connected network of the classification module. The two models are trained on labeled image dataset L_A. The test results of CNN-5 and Resnet34 on the test dataset T_A are shown in Table 1.

From Table 1, we can observe that the test result on Precision, Recall, and F1-score of CNN-5 is better than that of the Resnet34 pre-trained model. At the same time, because the CNN-5 model only uses five small convolutional kernels and one fully-connected layer, the training speed of the CNN-5 model is faster. We choose the CNN-5 model as the image classifier of co-training.

4.2 Text Classification based on TextRNN

When constructing a text classifier based on TextRNN, we use two bidirectional LSTM layers and one fully-connected layer. The dimension of word vectors is 300 and the hidden size of two bidirectional LSTM layers is 128. We do the text classification experiment both on OCR text extracted from the webpage screenshots and html text. The test results on the test dataset T_B are shown in Table 2.

From Table 2, we can see that the test result on Precision, Recall, and F1-score of using OCR text is better than that of using html text. The screenshots of webpage may contain some key text information that is not contained in



(a) CNN-5 (b) TextRNN

Figure 6: Performance of CNN-5 and TextRNN with different numbers of samples selected in one iteration

html. So it is necessary to extract text information from webpage screenshots by OCR. The text in the image has strong semantic features and is useful for identifying gambling websites.

4.3 The Number of Samples Selected in One Iteration

In this paper, the number of samples selected in one iteration is an important factor. In one iteration, p positive samples and q negative samples with confidence higher than threshold θ are selected. If we select a few samples, the efficiency of the algorithm is low. If we select a lot of samples, we may introduce some noisy samples with wrong labels. To obtain appropriate number p and q of samples selected, we perform the experiments with different numbers of samples. The experimental results are shown in Figure 6.

From Figure 6, we can observe that when the number p and q of samples selected in one iteration is 20, the performance of CNN-5 and TextRNN is poor, and the performance is better as the number increases. When the number is 40, CNN-5 obtain the best performance, when the number is 50, TextRNN obtain the best performance. After that, as the number increases, the performance of CNN-5 and TextRNN deteriorates. This phenomenon may be explained by the fact that when we select a small number of samples, we may lose a lot of useful samples. When we select a large number of samples, we may introduce some noisy samples with wrong labels, and the performance will decrease. When the number is within an appropriate range, we can avoid introducing noisy samples as much as possible and retain more useful samples for classification.

4.4 Co-training of CNN-5 and TextRNN

After the initial training of CNN-5 and TextRNN, the next step is to retrain the two classifiers with co-training. We set the prediction confidence threshold θ to 0.8, the number p and q of samples selected in one iteration to 50, and the iteration number g to 6. After six iterations of retraining on the unlabeled dataset U, the co-training of CNN-5 and TextRNN is completed, named Co-CNN-5 and Co-TextRNN. The test results of Co-CNN-5 and Co-TextRNN on the test datasets T_A and T_B are shown in Table 3.

Table 3: Results of CNN-5 and TextRNN after co-training

Evaluation metrics	Model	Gambling	Normal	Overall
Precision	CNN-5	0.8364	**0.9111**	0.8737
	Co-CNN-5	**0.8947**	0.9054	**0.9001**
	TextRNN	**0.9794**	0.9515	0.9654
	Co-TextRNN	0.9739	**0.9932**	**0.9835**
Recall	CNN-5	**0.9200**	0.8200	0.8700
	Co-CNN-5	0.9067	**0.8933**	**0.9000**
	TextRNN	0.9500	**0.9800**	0.9650
	Co-TextRNN	**0.9933**	0.9733	**0.9833**
F1	CNN-5	0.8762	0.8632	0.8697
	Co-CNN-5	**0.9007**	**0.8993**	**0.9000**
	TextRNN	0.9645	0.9655	0.9650
	Co-TextRNN	**0.9835**	**0.9832**	**0.9833**

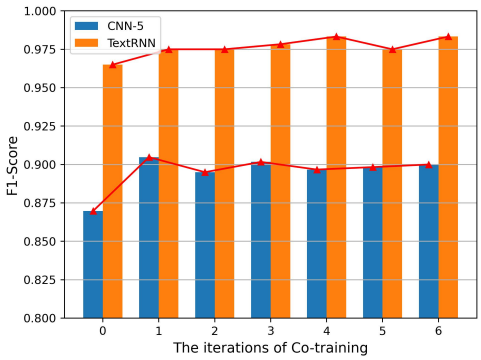


Figure 7: The performance of CNN-5 and TextRNN in different iterations of co-training

From Table 3, we can observe that after co-training, the test results on Precision, Recall, and F1-score of Co-CNN-5 and Co-TextRNN are better than that of CNN-5 and TextRNN before co-training. It indicates that the performance of the two classifiers are improved after co-training, and proves the effectiveness of the proposed method.

Meanwhile, the performance of CNN-5 and TextRNN in different iterations of co-training is shown in Figure 7. We can observe from Figure 7 that the F1-score of the CNN-5 and TextRNN shows an overall upward trend with the increase of the number of iterations. It infers that if the number of selected samples in one iteration is within an appropriate range, we can avoid introducing noisy samples and select more useful samples to improve the performance of classification models with co-training. We can observe that the performance of TextRNN is much better than CNN-5, this may be because the visual feature of webpage screenshots is too complex and the OCR text on the image has strong semantic features.

5 Related Work

URLs based methods extract features from URLs for classification. Fan et al. [1] identify illegal websites by detecting whether the unknown website contains illegal URL features. Garera et al. [2] study four different types of URL confusion structures used in phishing attacks, propose 18 URL features, and classify websites by logistic regression. Ma et al. [3] propose a website classification method based on URL features, which integrates the vocabulary features and domain name features of URLs. However, because the URLs contain insufficient features and information, the identification accuracy of URLs based methods is not high.

Webpage content-based methods extract content features from webpages for identification, such as HTML text, image, link, JavaScript code, and so on. Zhang et al. [4] extracted Chinese text from webpages and used text classification technology to classify webpages according to different themes. Li et al. [5] extract visual features from webpage screenshots to identify gambling and porn websites. Cernica et al. [6] propose a method that combines multiple techniques together with Computer Vision technique to detect phishing webpages. Jain et al. [7] propose a phishing website detection method by analyzing the hyperlinks in the webpage.

Mixed-feature-based methods combine different features to improve the accuracy of classification. Zhang et al. [8] propose a two-stage extreme learning machine for phishing website detection based on the mixed features of URL, web, and text content. Chen et al. [9] extract features from images and text in the webpages to detect gambling and porn websites.

6 Conclusion

In this paper, we propose a gambling websites identification method based on co-traning that extracts visual and semantic features of webpage screenshots and utilizes unlabeled data to improve the performance of classification models. We use OCR technique to extract text information in the webpage screenshots, the OCR text has strong semantic feature and is useful for identifying gambling websites. Then we construct an image classifier based on CNN and a text classifier based on TextRNN, the two classifiers are respectively trained from image view and text view. We retrain the image and text classifiers on unlabeled data with co-training algorithm. The experimental results indicate that the proposed method can effectively improve the performance of classifiers.

Acknowledgment

This work has been supported by the National Key R&D Program of China under Grant 2021YFB3100500.

References

[1] Y. Fan, T. Yang, Y. Wang, and G. Jiang, "Illegal website identification method based on url feature detection," *Computer Engineering*, 2018.

[2] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on Recurring malcode*, 2007, pp. 1–8.

[3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1245–1254.

[4] D. Zhang, "Research and implementation of content-oriented web page classification," *Nanjing University of Posts and Telecommunications, Nanjing, China*, 2017.

[5] L. Li, G. Gou, G. Xiong, Z. Cao, and Z. Li, "Identifying gambling and porn websites with image recognition," in *Pacific Rim Conference on Multimedia*. Springer, 2017, pp. 488–497.

[6] I. Cernica and N. Popescu, "Computer vision based framework for detecting phishing webpages," in *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2020, pp. 1–4.

[7] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 2015–2028, 2019.

[8] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage elm for phishing web pages detection using hybrid features," *World Wide Web*, vol. 20, no. 4, pp. 797–813, 2017.

[9] Y. Chen, R. Zheng, A. Zhou, S. Liao, and L. Liu, "Automatic detection of pornographic and gambling websites based on visual and textual content using a decision mechanism," *Sensors*, vol. 20, no. 14, p. 3989, 2020.

[10] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of ocr research and development," *IEEE Computer Society Press*, 1995.

[11] T. Technicolor, S. Related, T. Technicolor, and S. Related, "Imagenet classification with deep convolutional neural networks [50]."

[12] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[13] Z.-H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 415–439, 2010.

[14] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[15] Z.-H. Zhou, M. Li *et al.*, "Semi-supervised regression with co-training." in *IJCAI*, vol. 5, 2005, pp. 908–913.

[16] Y. Wang and T. Li, "Improving semi-supervised co-forest algorithm in evolving data streams," *Applied Intelligence*, vol. 48, no. 10, pp. 3248–3262, 2018.

[17] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, "Deep co-training for semi-supervised image recognition," in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 135–152.

[18] G. Katz, C. Caragea, and A. Shabtai, "Vertical ensemble co-training for text classification," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 2, pp. 1–23, 2017.

Social Information Popularity Prediction based on Heterogeneous Diffusion Attention Network

Xueqi Jia
College of Computer Science
Chongqing University
Chongqing, China
jiaxueqi99@126.com

Jiaxing Shang*
College of Computer Science
Chongqing University
Chongqing, China
shangjx@cqu.edu.cn

Linjiang Zheng
College of Computer Science
Chongqing University
Chongqing, China
zlj_cqu@cqu.edu.cn

Dajiang Liu
College of Computer Science
Chongqing University
Chongqing, China
liudj@cqu.edu.cn

Weiwei Cao
Key Laboratory of Flight Techniques and Flight Safety
Civil Aviation Flight University of China
Guanghan, China
ywcao@my.swjtu.edu.cn

Hong Sun
Flight Technology and Flight Safety Research Base
Civil Aviation Flight University of China
Guanghan, China
hanksun@263.net

***Abstract*—Information popularity prediction on social media platforms is a valuable and challenging issue. However, existing studies either neglect the correlation among different cascades, or lack a comprehensive consideration of user behavioral proximity and preference with respect to different messages. In this paper we propose a graph neural network-based framework named HeDAN (heterogeneous diffusion attention network), which comprehensively considers various factors affecting the information diffusion to predict the information popularity more accurately. Specifically, we first construct a heterogeneous diffusion graph with two types of nodes (*user* and *message*) and three types of relations (*Friendship*, *Interaction*, and *Interest*). Among them, *Friendship* reflects the strength of social relationship between users, *Interaction* reflects the behavioral proximity between users, and *Interest* reflects user preference to messages. Next, a graph neural network model with hierarchical attention mechanism is proposed to learn from these relations. Specifically, at the node-level, we utilize the graph attention network to learn the subgraph structure and generate the representations of nodes under each specific relationship. At the semantic-level, we distinguish the importance of different nodes in different relations via multi-head self-attention mechanism. Extensive experimental results on three datasets show the superior performance of our proposed model over the state-of-the-arts.**

***Index Terms*—Information popularity prediction, Graph neural network, Hierarchical attention, Social network analysis**

I. Introduction

Nowadays, social media platforms have greatly promoted the generation and dissemination of information, at the same time intensified the competition among different messages for users' attention. Among many of hot topics [1]–[3] related to social media analysis and mining, the theoretical and practical values of information popularity prediction have been widely recognized by both academia and industry. However, due to the openness of social media platforms and uncertainty of user behaviors, it is challenging to accurately predict the popularity of information on social media platforms.

Considering the importance of graph topology on information diffusion, graph representation-based methods have received more attention in recent years. Previous studies either focused on capturing the topology structure of a single diffusion graph [4]–[6] or mining the social dependencies between active and inactive users [7]–[9], which cannot directly exploit the correlations among different information cascades. However, the simultaneous consideration of all cascades can help to learn the interaction intimacy between users from their historical forwarding behaviors, which is helpful for accurately modeling information diffusion. Furthermore, messages attracting the same group of users are more likely to have similar popularity in the future, which means that establishing the direct links between messages and users can reflect the users' preferences to different messages, thereby benefiting the popularity prediction of messages. Therefore, this paper aims to comprehensively consider the role of social influence, interaction intimacy among users, and user preference to messages on information diffusion, so as to effectively capture predictive factors for more accurate information popularity prediction.

To this end, we propose a graph neural network-based framework named **HeDAN** (**He**terogeneous **D**iffusion **A**ttention **N**etwork), which utilizes a hierarchical attention mechanism to directly learn representations for both users and messages, so as to provide more accurate popularity prediction. Specifically, we first construct a heterogeneous diffusion graph with two types of nodes (*user* and *message*) and three types of relations (*Friendship*, *Interaction*, and *Interest*). Among them, *Friendship* refers to the underlying follower-followee relationships among users on social media platforms, which reflects the influence of users themselves from the perspective of social friendship. *Interaction* refers to the historical forwarding behaviors between users, which reflects the

*Corresponding author: Jiaxing Shang

This work was supported in part by: National Natural Science Foundation of China (Nos. 61966008, U2033213), Open Fund of Key Laboratory of Flight Techniques and Flight Safety, CAAC (Nos. FZ2021KF01, FZ2021KF14)

DOI reference number: 10.18293/SEKE2022-075

proximity among users from the perspective of user behaviors. *Interest* refers to the direct interactions between messages and users, which reflects the attractiveness of messages to users from the perspective of user preference. We creatively combine the above three types of relations to form a heterogeneous diffusion graph. Next, we propose a graph neural network model with hierarchical attention mechanism to learn from this heterogeneous diffusion graph. Specifically, at the node-level, we utilize graph attention network to learn the structure of the subgraphs according to the relationships and characterize the mutual importance of nodes. Then at the semantic-level, we utilize multi-head self-attention mechanism to distinguish the influence of different relationships and users on information diffusion, and finally fuse all kinds of influences to obtain the final representation vector for popularity prediction. Our main contributions and advantages are:

- We consider the correlation among different cascades and creatively construct a heterogeneous diffusion graph which contains three types of relations among users and messages to make information diffusion modeling more comprehensive.
- We propose a graph neural network model with hierarchical attention mechanism to learn from the heterogeneous diffusion graph for more accurate popularity prediction.
- Experimental results on three real-world datasets demonstrate the effectiveness of our proposed model, where the overall prediction errors are significantly reduced.

II. Related Work

In this paper, we will organize related works from the sequential representation-based methods and the graph representation-based methods.

A. Sequential representation-based methods

Sequential representation-based methods usually regard information cascades as dynamic time series and apply recurrent neural networks (RNN) to learn and model the diffusion process. DeepCas [4] utilized random walks to sample the cascade graphs to obtain the sequences of nodes as the input of the bidirectional gated recurrent unit (Bi-GRU). DeepHawkes [10] merged three crucial concepts of Hawkes process, i.e., user influence, self-exciting mechanism, and time decay effect, with RNN to make the modeling process more interpretable. DeepDiffuse [11] employed embedding technique and attention model to learn from the infection timestamp information.

B. Graph representation-based methods

With the development of graph neural networks (GNN) [12], graph representation-based information diffusion studies have received increasing attention in recent years. DeepInf [5] and DiffuseGNN [13] evaluated the social influence of the central user by predicting the user's state (active or inactive) based on the given r-ego network and neighbors' states. CasCN [6] sampled a cascade graph as a series of sequential subcascades and adopted a dynamic multi-directional GCN to learn structural information of cascades. DeepCon+Str [14] proposed two higher-order graphs with cascades as nodes based on content and structural proximity, and learned the higher-order graphs by random walks and semi-supervised language models. CoupledGNN [8] leveraged two specifically designed GNNs, one for node states and the other for influence spread, to model the cascading effect.

III. Methodology

In this section, we present the framework of our HeDAN (Heterogeneous Diffusion Attention Network) model, as illustrated in Fig.1. On the whole, HeDAN consists of the following four major components: (a) Heterogeneous diffusion graph construction module: which extracts the *Interaction* relations among users and *Interest* relations between users and messages from information cascades, and combines them with social relationships to construct a heterogeneous diffusion graph; (b) Node-level attention module: which utilizes graph attention networks to learn the graph structure under each specific relational subgraph, thereby generating node embeddings representing specific relationships; (c) Semantic-level attention module: which utilizes multi-head self-attention mechanism to distinguish the importance of different types nodes in different relationships, and fuse them into the final representation vector; (d) Prediction module: which transforms the final representation vector into the predicted popularity value via a multi-layer perceptron (MLP).

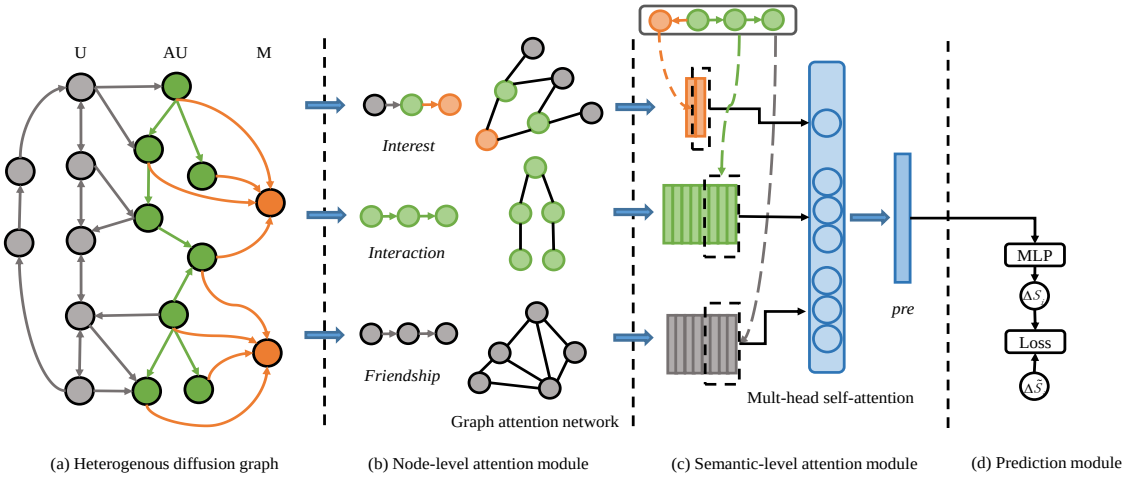
A. Heterogeneous diffusion graph construction module

We first define and construct a heterogeneous diffusion graph which contains two types of nodes (*user* and *message*) and three types of relations (*Friendship*, *Interaction*, and *Interest*). Fig.2 shows how to extract the corresponding relations from the cascade graphs (Fig.2(a)) and the global social graph (Fig.2(b)) to form a heterogeneous diffusion graph (Fig.2(c)).

Given the global social graph $\mathcal{G}_S$, a set of messages $\mathcal{M} = \{m_1, m_2, \cdots, m_d\}$ and the corresponding set of cascade graphs $\mathbf{G}_C = \{\mathcal{G}_{C_1}, \mathcal{G}_{C_2}, \cdots, \mathcal{G}_{C_d}\}$, the heterogeneous diffusion graph is defined as $\mathcal{G}_H = (\mathcal{V}_H, \mathcal{E}_H)$, where node set $\mathcal{V}_H = \mathcal{M} \cup \mathcal{V}_S \cup \mathcal{V}_{C_1} \cup \mathcal{V}_{C_2} \cup \cdots \cup \mathcal{V}_{C_d}$ is the set of all message nodes and user nodes. Each user node is associated with two states, active or inactive. If the user has participated in one of the messages, then it is active, otherwise it is inactive. The edge set $\mathcal{E}_H = \mathcal{E}_{F(u)} \cup \mathcal{E}_{I(u)} \cup \mathcal{E}_{I(m)}$ contains three subsets $\mathcal{E}_{F(u)}$, $\mathcal{E}_{I(u)}$ and $\mathcal{E}_{I(m)}$, where $\mathcal{E}_{F(u)} = \mathcal{E}_S$ is the *Friendship* edge set, $\mathcal{E}_{I(u)} = \mathcal{E}_{C_1} \cup \mathcal{E}_{C_2} \cup \cdots \cup \mathcal{E}_{C_d}$ is the *Interaction* edge set, and $\mathcal{E}_{I(m)} = \{(u_j, m_i) | u_j \in \mathcal{V}_{C_i}, m_i \in \mathcal{M}\}$ is the *Interest* edge set. As shown in Fig.2, the *Interest* edge set $\mathcal{E}_{I(m)}$ corresponding to m_1 in Fig.2(c) is $\{(u_1, m_1), (u_2, m_1), (u_3, m_1)\}$, where an edge (u, m) indicates that user u is interested in message m.

B. Node-level attention module

The purpose of this module is to model non-linear associations between nodes and generate the node representations under each relation type. As shown in Fig.1(b), this module generates three subgraphs from the original graph according



(a) Heterogenous diffusion graph (b) Node-level attention module (c) Semantic-level attention module (d) Prediction module

Fig. 1. The framework of HeDAN.

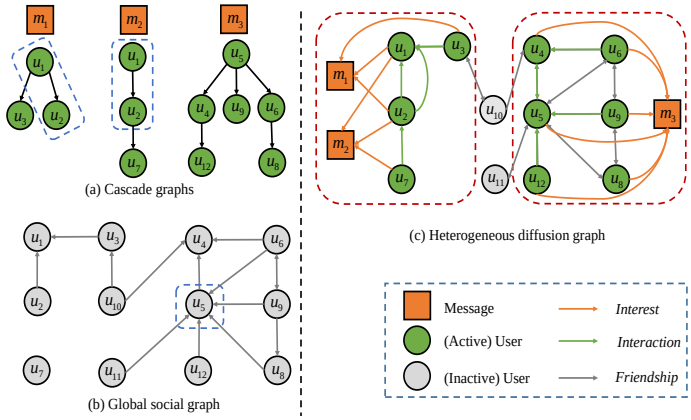


Fig. 2. An example of the heterogeneous diffusion graph. (a) An example of cascade graphs of message m_1, m_2, m_3(marked as orange squares). The edges denote that the user (marked as green circles) reposted a message from another user at a certain timestamp; (b) The global social graph consisting of follower-followee relationships between users; (c) The constructed heterogeneous diffusion graph, which includes two types of nodes (*user* and *message*) and three types of edges (*Friendship*, *Interaction*, and *Interest*). The green circles represent active user nodes, while the gray circles represent inactive user nodes.

to the three types of relationships, and then utilizes the graph attention network which incorporates the importance of neighbors to learn node representations on the subgraphs. The detailed process of this module is as follows:

1) Node feature transformation: Following the works [15] and [16] on heterogeneous graph representation learning, and considering that the feature spaces of message nodes and user nodes are different, we use transformation matrices to project both kinds of nodes into the same feature space. The projection process can be expressed as follows,

$$\mathbf{h}'^{(u)}_i = \mathbf{M}^{(u)} \cdot \mathbf{h}^{(u)}_i, \tag{1}$$

$$\mathbf{h}'^{(m)}_j = \mathbf{M}^{(m)} \cdot \mathbf{h}_j{}^{(m)}, \tag{2}$$

where $\mathbf{h}^{(u)}_i$ and $\mathbf{h}'^{(u)}_i$ are the original and projected features of the user node i, $\mathbf{h}^{(m)}_j$ and $\mathbf{h}'^{(m)}_j$ are the original and projected features of the message node j, $\mathbf{M}^{(u)} \in \mathbb{R}^{d_u \times d'}$ and $\mathbf{M}^{(m)} \in \mathbb{R}^{d_m \times d'}$ are the transformation matrices of user and message nodes respectively.

2) Friendship subgraph attention layer: We utilize the friendship subgraph attention layer to capture the friendship importance among users and obtain user representations based on friendship relations. The friendship subgraph $\mathcal{G}_{F(u)}$ is a bidirectional homogeneous subgraph generated by the edge-set $\mathcal{E}_{F(u)}$. $\mathcal{G}_{F(u)}$ is bidirectional because each social user plays two roles of sender and receiver in information diffusion. For example, if there is a following relationship between node B and node A, the edge (A, B) indicates that A is the sender and B is the receiver, while the edge (B, A) indicates that A is the receiver and B is the sender. Further, we adopt the graph attention layer to learn the importance $e^{F(u)}_{ij}$ on the subgraph $\mathcal{G}_{F(u)}$, which measures how sender j would contribute to receiver i on friendship. It can be formulated as follows,

$$e^{F(u)}_{ij} = \text{LeakyReLU}(\mathbf{w}^T_{F(u)} \cdot \left[\mathbf{h}'^{(u)}_i \parallel \mathbf{h}'^{(u)}_j\right]), \tag{3}$$

where $\mathbf{w}_{F(u)} \in \mathbb{R}^{2d'}$ are the parameterized attention vector for subgraph $\mathcal{G}_{F(u)}$ and $\parallel$ denotes the concatenate operation. Therefore, edge (A, B) and edge (B, A) can still learn different weight values, ie $e^{F(u)}_{ij} \neq e^{F(u)}_{ji}$.

Then we apply softmax function to obtain the normalized weight coefficient $\alpha^{F(u)}_{ij}$, which can be formulated as follows,

$$\alpha^{F(u)}_{ij} = softmax_j(e^{F(u)}_{ij}) = \frac{\exp(e^{F(u)}_{ij})}{\sum\limits_{k \in \mathcal{G}^{F(u)}_i} \exp(e^{F(u)}_{ik})}, \tag{4}$$

where $\mathcal{G}^{F(u)}_i$ is the first-order in-degree neighborhood of user i. For users with a large number of followers, due to its large in-degree value, the influence of each follower is lower on

average. For users with few followers but who are active in their own communities, the influence of their neighbors' connections is higher on average.

Finally, the embedding of node i in subgraph $\mathcal{G}_{F(u)}$ can be aggregated by the neighbors' projected features with the corresponding coefficients as follows,

$$\mathbf{z}_i^{F(u)} = \sigma(\sum_{j \in \mathcal{G}_i^{F(u)}} \alpha_{ij}^{F(u)} \cdot \mathbf{h'}_j^{(u)}), \tag{5}$$

where $\mathbf{z}_i^{F(u)}$ is the output of node i for subgraph $\mathcal{G}_{F(u)}$, and $\sigma(\cdot)$ is the activation function.

3) Interaction subgraph attention layer: We utilize the interaction subgraph attention layer to capture the interaction intimacy among activated users and obtain activated user representations based on interaction relations. Similar to $\mathcal{G}_{F(u)}$, we generate the interaction subgraph $\mathcal{G}_{I(u)}$ by the edge-set $\mathcal{E}_{I(u)}$. $\mathcal{E}_{I(u)}$ includes the forwarding relationship among activated users. We process the generated subgraph as a directed homogeneous graph $\mathcal{G}_{I(u)}$ and employ the graph attention layer to learn interaction attention and interaction-based user representations on $\mathcal{G}_{I(u)}$. Similar to that in friendship subgraph, the calculation formulas involved are as follows,

$$e_{ij}^{I(u)} = \text{LeakyReLU}(\mathbf{w}_{I(u)}^T \cdot \left[\mathbf{h'}_i^{(u)} \parallel \mathbf{h'}_j^{(u)}\right]), \tag{6}$$

$$\alpha_{ij}^{I(u)} = softmax_j(e_{ij}^{I(u)}) = \frac{\exp(e_{ij}^{I(u)})}{\sum_{k \in \mathcal{G}_i^{I(u)}} \exp(e_{ik}^{I(u)})}, \tag{7}$$

$$\mathbf{z}_i^{I(u)} = \sigma(\sum_{j \in \mathcal{G}_i^{I(u)}} \alpha_{ij}^{I(u)} \cdot \mathbf{h'}_j^{(u)}). \tag{8}$$

4) Interest subgraph attention layer: We utilize the interest subgraph attention layer to capture the user preferences for messages and to obtain message representations based on interest relations. When generating the interest subgraph, we consider two type of edges, one is the connections between the active users and the message, and the other is the virtual edges from other users and the message. A virtual edge means that if there is a reachable path of length 2 between an inactive user and a message, then a virtual edge is constructed for the inactive user as the source node and the message as the target node. Therefore, the interest subgraph $\mathcal{G}^{I(m)'}$ contains two directed bipartite subgraph, one is $\mathcal{G}_{IA}$ whose edges directly connect active users to messages, and the other is $\mathcal{G}_{IB}$ whose edges connect inactive users who are 2-hop away from the corresponding messages. Further, we train the graph attention network layer on $\mathcal{G}_{IA}$ and $\mathcal{G}_{IB}$ respectively, and finally get $\mathbf{z}_i^{I(m)}$. The formulas involved are as follows,

$$\alpha_{ij}^{IA} = \frac{\exp(\text{LeakyReLU}(\mathbf{w}_{IA}^T \left[\mathbf{h'}_i^{(m)} \parallel \mathbf{h'}_j^{(au)}\right]))}{\sum_{k \in \mathcal{G}_i^{IA}} \exp(\text{LeakyReLU}(\mathbf{w}_{IA}^T \left[\mathbf{h'}_i^{(m)} \parallel \mathbf{h'}_k^{(au)}\right]))}, \tag{9}$$

$$\alpha_{ij}^{IB} = \frac{\exp(\text{LeakyReLU}(\mathbf{w}_{IB}^T \left[\mathbf{h'}_i^{(m)} \parallel \mathbf{h'}_j^{(u)}\right]))}{\sum_{k \in \mathcal{G}_i^{IB}} \exp(\text{LeakyReLU}(\mathbf{w}_{IB}^T \left[\mathbf{h'}_i^{(m)} \parallel \mathbf{h'}_k^{(u)}\right]))}, \tag{10}$$

$$\mathbf{z}_i^{I(m)} = \sigma(\sum_{j \in \mathcal{G}_i^{IA}} \alpha_{ij}^{IA} \cdot \mathbf{h'}_j^{(au)} + \sum_{k \in \mathcal{G}_i^{IB}} \alpha_{ik}^{IB} \cdot \mathbf{h'}_k^{(u)} + \mathbf{h'}_i^{(m)}). \tag{11}$$

C. Semantic-level attention module

The goal of this module is to model the importance of different relationships on information diffusion, so as to obtain a vector representation that integrates the effects of various impacting factors. Through the learning of each relational subgraph by the node attention module, we obtain the friendship-based user representations $\mathbf{Z}^{F(u)}$, the interaction-based user representations $\mathbf{Z}^{I(u)}$, and the interest-based message representations $\mathbf{Z}^{I(m)}$. Now we apply the semantic attention to distinguish the importance of each relationship and generate the final representation by fusing the above representations. The specific process is as follows:

Suppose the popularity of message m is to be predicted, and the active user list within observation window is $[u_A, u_B, u_C]$. First, we query the message representation vector $\mathbf{v}^{I(m)}$ from matrix $\mathbf{Z}^{I(m)}$ according to the id of message m. Next, we query the friendship-based user representation vector list $[\mathbf{v}_A^{F(u)}, \mathbf{v}_B^{F(u)}, \mathbf{v}_C^{F(u)}]$ from matrix $\mathbf{Z}^{F(u)}$ and the interaction-based user representation vector list $[\mathbf{v}_A^{I(u)}, \mathbf{v}_B^{I(u)}, \mathbf{v}_C^{I(u)}]$ from matrix $\mathbf{Z}^{I(u)}$ according to the user id of the list $[u_A, u_B, u_C]$. Finally, we fuse the above vectors as $\tilde{\mathbf{V}} \in \mathbb{R}^{N \times d}$ for semantic attention learning, where N represents the number of vectors in the list. The specific implementation of semantic attention adopts the following multi-head self-attention mechanism,

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{12}$$

$$h_i = Attention(\tilde{\mathbf{V}}\mathbf{W}_i^{\mathbf{Q}}, \tilde{\mathbf{V}}\mathbf{W}_i^{\mathbf{K}}, \tilde{\mathbf{V}}\mathbf{W}_i^{\mathbf{V}}), \tag{13}$$

$$\mathbf{Y} = [h_1; h_2; ...; h_H]\mathbf{W}^{\mathbf{O}}, \tag{14}$$

$$\mathbf{pre} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n, \tag{15}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}^O \in \mathbb{R}^{Hd_k \times d_Q}$; $d_k = d/H$; H is the number of heads of attention module. $\mathbf{Y} \in \mathbb{R}^{N \times d_Q}$ represents the vector list after semantic fusion.

D. Prediction module

The last component of HeDAN is a multi-layer perceptron (MLP) with one final output unit. Given the representation vector pre_i, we calculate the popularity ΔS_i as:

$$\Delta S_i = MLP(\mathbf{pre}_i) \tag{16}$$

Our ultimate task is to predict the final cascade size of message m_i, which can be done by minimizing the following loss function:

$$loss(\Delta S_i, \Delta \tilde{S}_i) = \frac{1}{M}\sum_{i=1}^{M}(\log \Delta S_i - \log \Delta \tilde{S}_i)^2 \qquad (17)$$

where M is the number of messages, ΔS_i is the predicted popularity for message m_i, and $\Delta \tilde{S}$ is the ground truth.

IV. Experimental Evaluation

A. Dataset

We select three datasets containing both user social graphs and diffusion cascades [17] for experiments. The detailed statistics are presented in Tab.I.

TABLE I
STATISTICS OF THE TWITTER, DOUBAN, AND WEIBO DATASETS.

Datasets	Twitter	Douban	Weibo
#Users	12,627	23,123	2,000,000
#Links	309,631	348,280	12,822,901
#Cascades	3,442	10.662	22,767
#Train Casacdes	2,768	8,529	18,231
#Valid Cascades	345	1,067	2,265
#Test Cascades	344	1,066	2,271

B. Baseline & Evaluation Metric

1) Baseline: To evaluate the effectiveness of HeDAN, we select four methods from the existing deep learning-based methods for comparison. For the sequential representation-based methods, we select DeepCas [4] and DeepHawkes [10]. For the graph representation-based methods, we select DeepCon+Str [14] and CoupledGNN [8].

2) Evaluation metric: Following the existing works [4], [6], [10], we choose MSLE and mSLE as the evaluation metrics of the experiments.

C. Settings

For the baseline methods, the node embedding size of DeepCas, DeepHawkes and DeepCon+Str is set to 64, and all other hyperparameter settings of each model are set to their default values. For our model, the dimension of the hidden units is set to 64. For GAT, the number of heads in the multi-head attention is 8 and the dimension of each head is 8. For multi-head self-attention mechanism, we set the number of heads in the multi-head attention to 4. Our model is implemented by PyTorch. We employ the Adam optimizer with the learning rate set to 0.005 and the weight decay (L2 penalty) set to 0.001. We set the dropout rate to 0.6.

D. Results

1) Overall performance: Tab.II, Tab.III and Tab.IV show the performance of all methods on the three datasets, where the best results are highlighted.

TABLE II
EXPERIMENTAL RESULTS ON TWITTER DATASET.

Dataset	Twitter					
Observation time	1 hour		2 hours		3 hours	
Evaluation metric	MSLE	mSLE	MSLE	mSLE	MSLE	mSLE
DeepCas	1.3770	0.2788	1.3227	0.3092	1.3180	0.2992
DeepHawkes	0.9322	0.1710	0.8953	0.1615	0.8222	0.1552
DeepCon+Str	0.8847	0.1366	0.8521	0.1288	0.7297	0.1243
CoupledGNN	0.7867	0.1301	0.7660	0.1247	0.7045	0.1208
HeDAN	**0.7766**	**0.1263**	**0.7349**	**0.1211**	**0.6606**	**0.1127**

TABLE III
EXPERIMENTAL RESULTS ON DOUBAN DATASET.

Dataset	Douban					
Observation time	1 year		2 years		3 years	
Evaluation metric	MSLE	mSLE	MSLE	mSLE	MSLE	mSLE
DeepCas	1.1293	0.2564	1.0997	0.2369	0.9452	0.2408
DeepHawkes	0.8135	0.1820	0.7335	0.1788	0.7020	0.1665
DeepCon+Str	0.7026	0.1738	0.6854	0.1692	0.6663	0.1673
CoupledGNN	0.6330	0.1696	0.6262	0.1633	0.6035	0.1631
HeDAN	**0.6260**	**0.1676**	**0.6158**	**0.1618**	**0.5927**	**0.1618**

TABLE IV
EXPERIMENTAL RESULTS ON WEIBO DATASET.

Dataset	Weibo					
Observation time	2 hours		4 hours		6 hours	
Evaluation metric	MSLE	mSLE	MSLE	mSLE	MSLE	mSLE
DeepCas	2.2237	1.2260	2.2343	1.2394	2.2053	1.2102
DeepHawkes	1.5527	0.9886	1.5332	0.9727	2.5096	0.9662
DeepCon+Str	1.3724	0.9224	1.3522	0.9103	1.3042	0.8962
CoupledGNN	1.2228	0.7228	1.1055	0.6888	1.0134	0.6835
HeDAN	**1.1139**	**0.6905**	**1.0264**	**0.5707**	**0.9734**	**0.5664**

From Tab.II, Tab.III and Tab.IV, we can see that HeDAN outperforms the state-of-the-art methods by a significant margin. Specifically, we have the following observations: (1) The graph representation-based methods significantly outperform the sequence representation-based methods (over 10% improvement in MSLE on three datasets). This indicates that graph structural information learned by graph representation-based methods is useful for information modeling. (2) HeDAN outperforms DeepCon+Str, with MSLE and mSLE improved by nearly 20% on the Weibo dataset. Unlike DeepCon+Str which ignores fine-grained user-message interactions, HeDAN directly constructs the *Interaction* and *Interest* relationships, which reserves detailed information for users and cascades. Moreover, HeDAN uses the graph representation model such as GAT to learn the node representation, which better captures the internal structure of the cascades compared with the semi-supervised language model. (3) HeDAN outperforms CoupledGNN, with both MSLE and mSLE improved by nearly 10% on the Weibo dataset. This indicates that HeDAN considers co-processing of the cascades to capture the interactions between users and the relationship between cascades, which has a boosting effect in predicting the popularity of cascades.

2) Ablation experiments: To show the relative importance of each module in HeDAN, we perform a series of ablation studies over the key modules of the model. Fig.3 gives the overall performance on several variant methods of HeDAN. We can observe that: The performance of variants (1) (2) (3) shows that all three types of relations have a catalytic effect on information popularity prediction. Variant (4) demonstrates the effectiveness of GAT in node-level modules. The effectiveness of the multi-head self-attention mechanism in the semantic-level module is demonstrated through the variant (5).

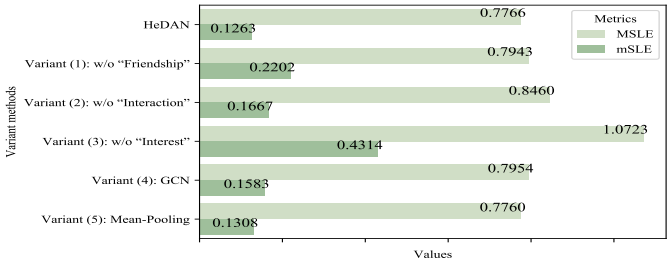


Fig. 3. Results of ablation experiments under a 1-hour observation time window on Twitter dataset.

3) Visualization: In this section, we utilize the t-SNE [18] algorithm to visualize the final prediction representations learned by HeDAN, as shown in Fig.4. We find a clear change in the popularity distribution in Fig.4 (weaker from left to right), which indicates that the latent representations learned by HeDAN are more expressive. Moreover, the distribution of datapoints in Fig.4 is aggregated rather than scattered, which reflects the characteristics of the regression problem.

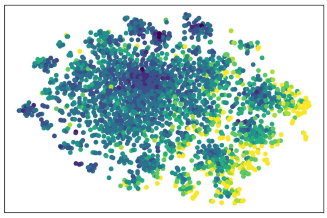


Fig. 4. A visualization of the final prediction representations of cascades on the Twitter dataset. Each datapoint represents a cascade. The darker the datapoint, the higher its popularity value. The locations of datapoints are obtained by performing t-SNE dimensionality reduction on their final prediction representations.

V. Conclusion

In this paper, we studied the information popularity prediction problem on social media platforms. To comprehensively consider various factors that affect information diffusion, we proposed a novel heterogeneous diffusion attention network model to characterize both the user and message representations through hierarchical attention. Specifically, we learned various subgraph structures through node-level attention, and creatively integrated the roles of friendship, user interaction and user preference through semantic-level attention. We conducted experiments on three real-world datasets. The experimental results indicate that our model achieved significant improvements over state-of-the-art models. As for future work, we will extend our model to fine-grained problems such as user-level diffusion behavior prediction. We will also consider model interpretability in our future work.

References

[1] S. Bhattacharya, K. Gaurav, and S. Ghosh, "Viral marketing on social networks: An epidemiological perspective," *Physica A: Statistical Mechanics and its Applications*, vol. 525, pp. 478–490, 2019.

[2] F. Ducci, M. Kraus, and S. Feuerriegel, "Cascade-lstm: A tree-structured neural classifier for detecting misinformation cascades," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2666–2676.

[3] C. Huang, H. Xu, Y. Xu, P. Dai, L. Xiao, M. Lu, L. Bo, H. Xing, X. Lai, and Y. Ye, "Knowledge-aware coupled graph neural network for social recommendation," in *35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[4] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 577–586.

[5] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2110–2119.

[6] X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong, and F. Zhang, "Information diffusion prediction via recurrent cascades convolution," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 770–781.

[7] Z. Wang, C. Chen, and W. Li, "A sequential neural information diffusion model with structure attention," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1795–1798.

[8] Q. Cao, H. Shen, J. Gao, B. Wei, and X. Cheng, "Popularity prediction on social platforms with coupled graph neural networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 70–78.

[9] S. Molaei, H. Zare, and H. Veisi, "Deep learning approach on information diffusion in heterogeneous networks," *Knowledge-Based Systems*, vol. 189, p. 105153, 2020.

[10] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng, "Deephawkes: Bridging the gap between prediction and understanding of information cascades," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1149–1158.

[11] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan, "Deepdiffuse: Predicting the'who'and'when'in cascades," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1055–1060.

[12] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[13] F. Zhang, J. Tang, X. Liu, Z. Hou, Y. Dong, J. Zhang, X. Liu, R. Xie, K. Zhuang, X. Zhang *et al.*, "Understanding wechat user preferences and "wow" diffusion," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[14] X. Feng, Q. Zhao, and Z. Liu, "Prediction of information cascades via content and structure integrated whole graph embedding." IJCAI, 2019.

[15] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[16] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[17] F. Zhou, X. Xu, G. Trajcevski, and K. Zhang, "A survey of information cascade analysis: Models, predictions, and recent advances," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–36, 2021.

[18] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

Context-Aware Model for Mining User Intentions from App Reviews

Jinwei Lu[†], Yimin Wu[†§], Jiayan Pei[‡], Zishan Qin[†], Shizhao Huang[‡] and Chao Deng[§]
[†]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[‡]School of Software Engineering, South China University of Technology, Guangzhou, China
[§]School of Computer Science and Engineering, Guangdong Ocean University at yangjiang, Yangjiang, China
cskilljl@mail.scut.edu.cn, csymwu@scut.edu.cn, seasensio@mail.scut.edu.cn,
csqzs@mail.scut.edu.cn, se_hsz@mail.scut.edu.cn, dengchao@gdou.edu.cn

***Abstract*—Due to the highly competitive and dynamic mobile application (app) market, app developers need to release new versions regularly to improve existing features and provide new features for users. To accomplish the maintenance and evolution tasks more effectively and efficiently, app developers should collect and analyze user reviews, which contain a rich source of information from user perspective. Although there are many approaches based on intention mining that can automatically predict the intention of reviews for better understanding valuable information, those approaches are limited since contextual information of the whole review text may be lost. In this paper, we propose Mining Intention from App Reviews (MIAR), a novel deep learning model to predict the intention of app reviews automatically. We adopt a Contextual Feature Extractor to capture the context semantic information and fuse it with the local feature through a fusion mechanism. The experiment results demonstrate that MIAR has made significant improvement over the baseline approaches in Precision, Recall, and F1-score evaluation metrics, achieving state-of-the-art performance in this task. Our model also performs well in other intention mining tasks, proving its generalization ability and robustness.**

***Keywords*—Software Maintenance; Deep Learning; App Review; Intention Mining**

I. Introduction

With the highly competitive and dynamic app market, it is essential for app developers to regularly release new versions to fix bugs and offer new features to users. In order to understanding the changing user needs which are related to app maintenance and evolution, many app stores offer a review module to users, which provide a communication channel to users and developers [1]. However, manually processing such user reviews is a big challenging. Many useless reviews do not express any valuable information, or unstructured reviews use non-technical language, making review processing task become time-consuming and error-prone. Therefore, many researchers have developed a variety of approaches to automatically identifying relevant user reviews [2] [3], or by clustering and prioritizing user reviews to find the most crucial topics [4].

Apart from previous approaches which are helpful to cope with large amounts of user review, it is also important for app developers to understand the intention that users expressed implicitly in reviews [5], which could provide valuable information in detecting relevant content to accomplish several maintenance and evolution tasks. To help identify user intentions more accurately, many researchers leveraged a process called intention mining to analyze and filter user review sentences. While these researches have performed well in classifying app reviews into different user intentions, their methods based on linguistic patterns matching may waste a lot of time and manual inspection in pattern identification. Moreover, it is still difficult to accomplish this task when reviews are unstructured or intentions are implicit, which will be hard for these methods to distinguish the relevant components and patterns for classification.

In recent years, some researchers adopt deep learning to overcome this shortcoming. With the strong non-linear fitting ability, deep learning can effectively extract semantic information. Some researchers have adopted deep learning to identify intentions from various communication channels [6] [7]. Taking Huang *et al.*'s [7] work as an example, their approach builds a convolutional neural network (CNN) to classify sentences from issue tracking systems. However, they only adopt neural network to learn local representation of sentence, which ignores the contextual information of the text.

This paper conducts research based on Huang *et al.*'s work. In order to leverage intention mining to accomplish review mining task, we propose a novel model, named Mining Intention from App Reviews, which can detect intentions via neural network. To better understand the contextual information in the review, we build a context-aware model based on BiLSTM [8] that can deeply learn the global representation. To evaluate the proposed model, we conduct experiments on the review dataset to compare with state-of-the-art method, and results demonstrate the superior performance of our proposed method.

The main contributions of this paper are as follows:

- We propose a novel intention mining model for app review classification task, which does not rely on the linguistic patterns and greatly reduce the manual effort.
- We implement a contextual feature extractor to capture the global feature for better gaining the context sematic information and understanding user intentions.
- We evaluate our approach on several relevant datasets, and the result shows that our model not only outperforms previous methods in app review domain, but also have a good performance in other software engineering domains.

DOI reference number: 10.18293/SEKE2022-084

The rest of the paper is organized as follows. Section 2 briefly presents the related work of our study. Section 3 introduces the overall framework and technical details of our approach. Section 4 describes the experimental settings and presents the experiment results. Finally, Section 5 concludes the paper and outlines future work.

II. Related Work

App review is essential to app developers, as it contains valuable insights that can help successfully accomplish app maintenance and evolution tasks. Pagano and Maalej [1] identified 17 topics in user feedback by manually investigating the content of selected user reviews, which included the topics of *feature request* or *bug report* that could be mined for requirements-related information.

Many previous researchers used machine learning methods based on linguistic rules or heuristics patterns to extract such information. Iacob and Harrison [2] extracted feature requests from app reviews utilizing linguistic rules and used Latent Dirichlet Allocation (LDA) to group them. AR-Miner, which was proposed by Chen *et al.* [4], also employed LDA to group informative reviews. Maalej and Nabil [3] generated a list of keywords to be applied for the classification task. Then, they applied various machine learning methods to classify reviews. Panichella *et al.* [5] hypothesized that understanding the intention in a review has an important role in extracting useful information for developers. Therefore, they leveraged intention mining, which was proposed by Di Sorbo *et al.* [9], to catch useful contents. They merged three techniques to mining user intention for classify app reviews into the categories which are relevant to user intention. After that, they proposed AR-Doc [10], which is based on J48 algorithm, to use linguistic patterns for classification. In a later work, Di Sorbo *et al.* [11] proposed SURF to summarize app review for software change recommendation. Palomba *et al.* [12] proposed ChangeAdvisor to extract the intention of reviews to analyze potential app evolution. These tools leveraged classifier proposed by Panichella *et al.* [5], which means that intention mining can be a fundamental component to support complicated tasks. In order to minimize the manual effort of relevant pattern tagging, Di Sorbo [13] proposed NEON to automatically mine linguistic rules for review analysis and classification. However, these methods based on syntax analyzing or linguistic patterns matching limit the ability to extract semantic information from reviews, which means that the classifier could not understand the implicit intentions and just use the explicit feature to identify the category of app reviews. Therefore, to solve this problem, we leverage deep learning to model high-level abstractions in data by building neural networks with multiple layers.

In recent years, some researchers have explored the possibility of applying deep learning for intention mining. Stanik *et al.* [14] used a simple CNN-based model to classify user feedback for software development. Huang *et al.* [7] also proposed a CNN-based approach, which improves Di Sorbo *et al.*'s approach [9] and the other automated sentence classification approaches by a substantial margin. However, in their work, deep learning approaches were based on the local feature extraction model, which learns the representation of the receptive field and only calculates the relevance between adjacent n-gram elements. Hence these methods can not sufficiently consider the contextual information of the whole text, which is essential to predicting the intention of app review. Therefore, based on deep learning, we adopt the global feature extraction mechanism to learn the contextual information, hoping to achieve better performance in the review intention mining task.

III. Approach

According to Panichella *et al.* [5], user intention categories of app review can be defined as the following four classes:

- *Information Giving (IG)* : sentences that inform or update users or developers about an aspect related to the app.
- *Information Seeking (IS)* : sentences related to attempts to obtain information or help from other users or developers.
- *Feature Request (FR)* : sentences expressing ideas, suggestions or needs for improving or enhancing the app or its functionalities.
- *Problem Discovery (PD)* : sentences describing issues with the app or unexpected behaviors.

The intention of a review is predicted to be one of the four classes mentioned above. We use raw text sentences of a review as the input sequence. Suppose the input text sequence is $R = \{w_1, w_2, \ldots, w_n\}$,where *n* is the sequence length.

Figure 1 presents the overall framework of MIAR, which is mainly composed of the following five modules: (1) *Word Embedding Layer*, (2) *Local Feature Extractor*, (3) *Contextual Feature Extractor*, (4) *Fusion Layer*, and (5) *Prediction Layer*.

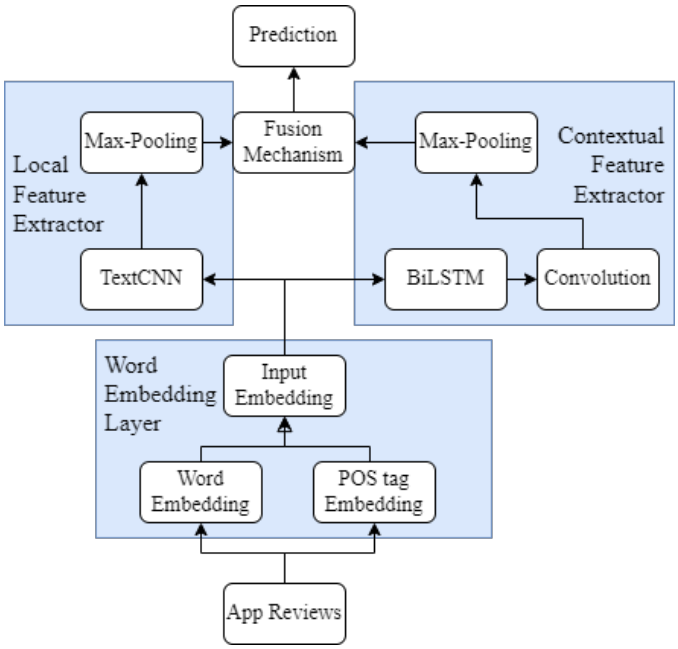


Fig. 1. The framework of MIAR

A. Word Embedding Layer

In this Layer, we leverage Word Embedding technique to transform words into the corresponding vector representations. Each word w_i in the input sequence is transformed into a vector representation $x_i \in \mathbb{R}^d$ through the pre-trained word embeddings. In our work, we use the pre-trained *GloVe* word embeddings with 300 dimensions [15]. Then, we train word vectors to obtain word embeddings. Of course, all kinds of word embedding methods can be employed in this process.

Moreover, inspired by previous work [16], making good use of POS tag can benefit semantic understanding by extracting explicit lexical information. Therefore, we add POS tag information into word vector to augment its ability of feature representation. Specifically, each type of POS tag is initialized as a random vector with uniform distribution and optimized during training. Hence, each word can be represented as:

$$x_i = [x_i^e \oplus x_i^p] \tag{1}$$

Where $\oplus$ is the concatenation operator, x_i^e and x_i^p denotes the corresponding word embedding and the embedding of the POS tag of the word, respectively.

Therefore, the review sequence containing *n* words can be converted to corresponding matrix representation, which is the input of the two feature extractors of the model:

$$Rx = x_1 \oplus x_2 \oplus \ldots \oplus x_n \tag{2}$$

B. Local Feature Extractor

In this module, we adopt TextCNN [17], which is a type of CNN for sentence classification, to extract the local feature in the text sequence. The convolutional layer receives the matrix representation Rx and performs convolution operation on it using different filters. Each filter is also a matrix, denoted as F, having the same width as the matrix R. The purpose of each filter with height *f* is to capture the semantic feature of each n-gram sequence in the sentence through a convolution operation, which is computed as follows:

$$C_i = F \cdot R_{i:i+f-1} + b_i, \forall i \in \{1, 2, \ldots, n-f+1\} \tag{3}$$

Where $R_{i:i+f-1}$ represents the sub-matrix of R from the *i* th row to the (*i+n-1*) th row, which represents the vectors of *f* continuous words (n-gram), and b_i is a bias value.

The *j* th filter is applied repeatedly to each n-grams in the sentence with convolution operation and produces a vector:

$$F_j = [C_1, C_2, \ldots, C_{n-f+1}], \forall j \in \{1, 2, \ldots, m\} \tag{4}$$

Where *n* is the height of matrix R_x. In order to help the extractor to learn enough semantic features in different granularities, multiple filters with various heights are used.

After performing the convolution operation, a pooling layer is applied to reduce the number of parameters and the computation cost. Specifically, the pooling layer applies a 1-max pooling function to the vector F_j received from each filter. Then outputs of *m* filters are concatenated as a high-level feature vector, which can catch semantic features of different n-grams in the input, representing the local feature vector *LF*:

$$LF = maxpooling([F_1; F_2; \ldots; F_m]) \tag{5}$$

C. Contextual Feature Extractor

To solve the problem of Local Feature Extractor that is not sensitive to the sequential information which is essential to understand the semantic relation and implicit intention, we build a Contextual Feature Extractor to extract the contextual information and generate the global representation of the input sequence. Here, we adopt BiLSTM to incorporate the contextual information into the original representation of each token in input sequence. BiLSTM is composed of a forward and a backward LSTM. Through its three gate structures, LSTM can solve the long-term dependence information very well, which means that bi-directional semantic dependencies within the review can be well captured. We concatenate the outputs of the two LSTM to generate the augmented vector of a token, which incorporates the contextual information into the token representation. This process can be represented as:

$$\overrightarrow{h}_i = LSTM(\overrightarrow{h}_{i-1}, x_i), \forall i \in \{1, 2, \ldots, n\} \tag{6}$$

$$H_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \forall i \in \{1, 2, \ldots, n\} \tag{7}$$

Where $\overrightarrow{h}_i$ is the hidden state of the forward LSTM in the time step *i*, $\overleftarrow{h}_i$ represents the backward, x_i is the input of LSTM in the time step i, and H_i is the contextual representation.

To condense the rich information extracted by BiLSTM, we add a convolutional layer to abstract the contextual feature. The feature vector transferred from BiLSTM does not contain sequential information incorporated into representation, which is suitable for convolutional network to perform feature condensing. Then a max-pooling layer is leveraged to conduct feature dimension reduction and generate the global representation. This global feature vector contains high concentration contextual information that can help mining intention better. The global feature vector *GF* can be calculated as follow:

$$HC_i = F_h \cdot H_{i:i+f-1} + b_i, \forall i \in \{1, 2, \ldots, n-f+1\} \tag{8}$$

$$F_j = [HC_1, HC_2, \ldots, HC_{n-f+1}], \forall j \in \{1, 2, \ldots, m\} \tag{9}$$

$$GF = maxpooling([F_1; F_2; \ldots; F_m]) \tag{10}$$

Where $H_{i:i+f-1} = [H_i, \ldots, H_{i+f-1}]$, *f* is filter size, *m* is the number of filters, and b_i is a bias value. Then the final representation is passed to the fusion layer for feature fusion.

D. Fusion Layer

In the Fusion Layer, *LF* and *GF* are integrated to generate the intention representation. We fuse the features as follow:

$$\tilde{X}_i^1 = \alpha LF_i + (1-\alpha) GF_i \tag{11}$$

$$\tilde{X}_i^2 = \beta LF_i - (1-\beta) GF_i \tag{12}$$

$$\tilde{X}_i^3 = \gamma GF_i - (1-\gamma) LF_i \tag{13}$$

Where $+$ represents feature augment computation, which highlights the similarity between two vectors, and $-$ denotes the difference between two vectors. α, β, and γ are the weighting factors which can be tuned for controlling the fusion degree

of each feature. We concatenate the results obtained by the above three methods and input them into another single-layer feed-forward network F to compute the output:

$$\tilde{X}_i^1 = F([\tilde{X}_i^1; \tilde{X}_i^2; \tilde{X}_i^3]) \tag{14}$$

Where $[;]$ refers to the concatenation operation. Then the final feature representation is passed to the Prediction Layer.

E. Prediction Layer

In the Prediction Layer, for vectors obtained from last layer, we use a multi-layer feed-forward network L to get feature vectors. Then a softmax function is applied to normalize the values so that the output vector can represent the probability of the input sequence belonging to one specific category:

$$\sigma(V_i) = \frac{e^{V_i}}{\sum_{j=1}^{K} e^{V_j}}, \forall i \in 1, 2, \ldots, K \tag{15}$$

Finally, we use the cross-entropy function to measure the loss between the prediction result and the ground truth:

$$Loss = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{16}$$

IV. Experiment

In this section, we conduct some experiments for answering the following research questions.

A. Research Questions

RQ1: How effective is MIAR for predicting user intention of different categories in app reviews?

MIAR adopts deep learning to automatically mining review intention based on two feature extractors and the fusion mechanism, which is much different from the previous works. To investigate the effectiveness of our approach, we compare the performance of MIAR with the baseline from Panichella *et al.*'s work [5]. Moreover, we also conduct experiments with other three intention mining models including AR-Doc [10], DECA [9], and Huang *et al.*'s CNN-based model [7]. To present results more accurately, we keep results of all models to three decimal places.

RQ2: How much influence do the techniques or modules we proposed contribute to the improvement of MIAR?

Three important techniques we proposed, including POS tag embedding (POS), Contextual Feature Extractor (CFE), and Fusion Layer (FL), could help MIAR to capture semantic information and identify intention from app reviews better. To evaluate their contributions, we conduct an ablation study to demonstrate. We take turns to remove one of the three techniques and compare the revised model with the original model on the *F1-score*. Specifically, when we remove CFE, we must remove FL simultaneously, since the fusion mechanism needs global feature generated by CFE to perform the fusion process. Thus, we perform the following ablation studies, consisting in (1) only removing POS; (2) only removing FL and replacing by concatenation; and (3) removing the CFE.

RQ3: Does MIAR work well in intention mining tasks from other software engineering domains?

TABLE I
Details of Datasets

Domain	IG	IS	FR	PD	Other	Total
Review	583	101	218	488	31	1421
Issue	1,328	962	536	762	397	3,985
Email	167	264	187	170	0	788

To explore the generalization ability of MIAR, we apply MIAR to emails mining task [9] and issue mining task [7] for comparison. Following Huang *et al.*'s work, we conduct the following studies, including (1) only using issue dataset (Issue); (2) only using email dataset (Email); (3) using issue dataset as training set and email dataset as test set (I to E); (4) using email dataset as training set and issue dataset as test set (E to I). The results of baseline models are from paper [7].

B. Dataset

We carry out experiments on the review dataset built by Panichella *et al.* [5]. They sampled 1421 review sentences out of 7696 reviews and manually labeled the sample according to the categories of their intention taxonomy, which includes four intention classes and *Other* class. We use 3-folds cross-validation to carry out our experiments on this dataset.

Moreover, to evaluate the generalization ability of MIAR, we also run experiments on two intention mining datasets proposed by Di Sorbo *et al.* [9] and Huang *et al.* [7]. We also carry out 3-folds cross-validation as Huang *et al.* [7] to evaluate the performance of MIAR. The detailed data of these three datasets is presented in Table I.

C. Evaluation Metric

We use the same evaluation metrics as Huang *et al.* [7] to evaluate MIAR's performance.

Precision represents the proportion of samples predicted to be positive that are truly positive samples.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{17}$$

Recall represents the proportion that the positive samples are predicted to be positive correctly.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{18}$$

F1-score is the weighted average of *Precision* and *Recall*. This metric takes into account both the *Precision* and *Recall* of the model. In the multi-class classification task, *F1-score* are computed for each class and then averaged via arithmetic mean to get *Macro-F1*.

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{19}$$

D. Implementation Details

MIAR is implemented based on the PyTorch [18] framework, experimented on an Nvidia 1080Ti GPU. For the Local Feature Extractor, we set two different kernel sizes, which are 2 and 3, respectively, and 100 feature maps for each kernel.

TABLE II
MAIN RESULT

	IG			IS			FR			PD					
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg-P	Avg-R	Macro-F1
Baseline	0.680	**0.904**	0.776	0.712	0.684	0.698	**0.704**	0.225	0.341	**0.875**	0.776	0.823	0.743	0.647	0.659
DECA	0.327	0.730	0.451	0.723	0.640	0.679	0.516	0.281	0.364	0.733	0.810	0.769	0.575	0.615	0.566
Huang	0.677	0.788	0.728	0.793	0.396	0.528	0.512	0.401	0.450	0.745	0.717	0.731	0.682	0.575	0.609
AR-Doc	0.456	0.747	0.566	0.683	0.633	0.657	0.625	0.385	0.476	0.751	0.808	0.779	0.629	0.643	0.620
MIAR	**0.750**	0.880	**0.810**	**0.871**	**0.692**	**0.772**	0.578	**0.646**	**0.610**	0.829	**0.836**	**0.833**	**0.757**	**0.764**	**0.756**

TABLE III
COMPARING THE F1-SCORE OF MIAR AND THE REVISED MODELS

	IG	IS	FR	PD	Macro-F1
MIAR	**0.810**	**0.772**	**0.610**	**0.833**	**0.756**
MIAR(-POS)	0.761	0.757	0.570	0.787	0.719
MIAR(- FL)	0.748	0.724	0.531	0.752	0.689
MIAR(- CFE -FL)	0.703	0.664	0.488	0.728	0.646

TABLE IV
PERFORMANCE OF DIFFERENT MODELS IN OTHER DATASETS

		IG	IS	FR	PD	Macro-F1
Issue	DECA	0.293	0.511	0.420	0.601	0.456
	Huang	0.805	0.904	0.794	0.820	0.831
	MIAR	**0.842**	**0.980**	**0.846**	**0.829**	**0.874**
Email	DECA	0.743	0.874	0.789	0.879	0.821
	Huang	0.785	0.883	0.793	**0.890**	0.838
	MIAR	**0.804**	**0.954**	**0.842**	0.818	**0.854**
I to E	DECA	**0.743**	0.874	**0.789**	**0.879**	**0.821**
	Huang	0.562	0.808	0.579	0.760	0.678
	MIAR	0.659	**0.943**	0.693	0.811	0.776
E to I	DECA	0.293	0.511	0.420	0.601	0.456
	Huang	0.487	0.678	0.579	0.520	0.566
	MIAR	**0.684**	**0.890**	**0.670**	**0.746**	**0.747**

For the Contextual Feature Extractor, we use BiLSTM with 700 units as the encoder. Adam optimizer [19] with an initial learning rate of 0.001 is applied. We release the source code of MIAR and hope to facilitate future researches.[1]

E. Result

RQ1: How effective is MIAR for predicting user intention of different categories in app reviews?

Table II presents the experiment results. The best results are highlighted in bold. We can see that MIAR achieves the best results for all the four intention classes, with an average of 75.7%, 75.4%, and 75.6% in *precision*, *recall*, and *F1-score*. Although Baseline method has a better performance of *precision* in some classes, the *F1-score* of this model is approximately 9.7% below than that of MIAR, which is limited by its low *recall*, which is 11.7% below than MIAR. This result indicates that using the approach based on linguistic pattern can predict intentions precisely, but this method may be confused by some ambiguous patterns which can appear in different intentions. Moreover, these tools relies heavily on manual effort. The lack of relevant linguistic pattern for matching can seriously degrade the performance of these tools. In contrast, deep learning methods, which do not rely on the fixed pattern, can model the high-level abstractions of reviews to understand the whole sentence and achieve higher score.

The result of *FR* is lower than other classes, which indicates that the semantic information of this class is more confused for understanding, or the expression is more implicit. So that Baseline and DECA are difficult to identify patterns for classification, which leads to a poor result (34.1% and 36.4%). This result reflect that this class limits the overall performance of all approaches included MIAR, which still have a major improvement of *F1-score* (at least 13.4%) compared with other approaches. For the remaining three classes, MIAR's improvement is also significant $(8.2\%-24.4\%)$ compared with Huang *et al.*'s approach. This result indicates that extracting contextual features and semantic relations is essential for understanding and identifying user intentions, especially in the noisy and informal communication environment.

RQ2: How much influence do the techniques or modules we proposed contribute to the improvement of MIAR?

The experimental results are shown in Table III. After removing POS or FL, respectively, the revised models' performance decreases in some degree ($3.7\%and6.7\%$, in terms of *Macro-F1*, respectively). However, after removing the CFE, the model's performance decreases more obviously $(11.0\%$ in *Macro-F1*), especially in predicting the *Feature Request* class (12.2% in terms of *F1-score*). This experiment results prove the importance of the CFE, which plays an essential role in predicting review intention. Certainly, POS and FL also improve our model's performance.

RQ3: Does MIAR work well in intention mining tasks from other software engineering domains?

Table IV presents the experiment results. For Issue, Email and E to I, MIAR outperforms other models, which include the state-of-the-art model on these datasets. For I to E, MIAR has a performance degradation, due to the poor performance in *IG* and *FR*, which are also the bottleneck of other models. This experimental result may cause by the gap between different communication channels. The discussion from issue is more similar to app review in the statement, which has more verbal and indirect expression that understanding the contextual information is more important. Moreover, MIAR achieves better results in all the four experiments than Huang *et al.*'s model,

[1]Our model is openly available in https://anonymous.4open.science/r/MIAR/

TABLE V
CASE STUDY

Case	Review	AR-Doc	Huang	MIAR	Label
1	**Crashing Bug** Normally I will be able to find a work Around but I couldn't get a ROM to run.	PD	PD	FR	FR
2	And I'm also not sure how to search for hybrid cards am I **missing something** here	PD	PD	IS	IS

which shows that MIAR has good generalization ability that can be applied to other software engineering domains.

F. Case Study

To investigate how our architecture makes a difference in details, we visualize two examples from different classes in Table V. The most important phrases extracted by CNN-based architectures are highlighted in bold, and we underline the important contextual information extracted by CFE.

In case (1), AR-Doc captures the linguistic pattern: "*someone get something*", and this pattern generally apply to express some problems. So that the linguistic pattern belongs to *PD*, leading to the wrong predicting. Huang *et al.*'s approach extracts the most important phrase "*Crashing Bug*", which generally indicates some bugs. Thus their model misclassifies this review into *PD*. In contrast, MIAR not only captures the important local feature"*Crashing Bug*", but also extracts some contextual information: "*couldn't get*" and "*run*", which belong to long distance dependency and could be ignored by Local Feature Extractor. These contextual features can provide more semantic information for predicting the correct label.

In case (2), both AR-Doc and Huang *et al.*'s approach are disturbed by the influential local feature "*missing something*", which is inclined to report some errors appeared in the app, while MIAR can consider the helpful contextual feature "*not sure*" and "*search for*", which is effective on understanding the implicit intention and predicting *IS* correctly. In short, with the help of CFE, MIAR can extract more useful information and provide to the classifier for efficiently mining intentions.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a deep learning model MIAR based on the feature fusion mechanism to predict the user intention from app review, which can reduce the manual effort and better help developers obtain useful information for software maintenance and evolution. The experiment results show MIAR's effectiveness and consistency in predicting review intention, outperforming some baseline models in previous works. Moreover, in some other intention mining tasks, MIAR can also achieve state-of-the-art performance, which proves its generalization ability. We will explore the application that identify the intention of other written communication channels from software engineer domain in future work.

REFERENCES

[1] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*. IEEE, 2013, pp. 125–134.

[2] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *2013 10th working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 41–44.

[3] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015, pp. 116–125.

[4] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 767–778.

[5] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 281–290.

[6] M. Haering, C. Stanik, and W. Maalej, "Automatically matching bug reports with related app reviews," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 970–981.

[7] Q. Huang, X. Xia, D. Lo, and G. C. Murphy, "Automating intention mining," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1098–1119, 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. C. Gall, "Development emails content analyzer: Intention mining in developer discussions (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 12–23.

[10] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "Ardoc: App reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, 2016, pp. 1023–1027.

[11] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 499–510.

[12] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 106–117.

[13] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. D. Penta, G. Canfora, and H. C. Gall, "Exploiting natural language structures in software informal documentation," *IEEE Transactions on Software Engineering*, vol. 47, no. 8, pp. 1587–1604, 2021.

[14] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 220–226.

[15] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[16] L. Shi, M. Xing, M. Li, Y. Wang, S. Li, and Q. Wang, "Detection of hidden feature requests from massive chat messages via deep siamese network," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 641–653.

[17] Y. Kim, "Convolutional neural networks for sentence classification," *Eprint Arxiv*, 2014.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

COAT: A Music Recommendation Model based on Chord Progression and Attention Mechanisms

Weite Feng, Tong Li*, Zhen Yang
Beijing University of Technology
*litong@bjut.edu.cn

***Abstract*—Recently, efforts have been made to explore introducing music content into deep learning-based music recommendation systems. In previous research, with reference to tasks such as speech recognition, music content is often fed into recommendation models as low-level audio features, such as the Mel-frequency cepstral coefficients. However, unlike tasks such as speech recognition, the audio of music often contains multiple sound sources. Hence, low-level time-domain-based or frequency-domain-based audio features may not represent the music content properly, limiting the recommendation algorithm's performance. To address this problem, we propose a music recommendation model based on chord progressions and attention mechanisms. In this model, music content is represented as chord progressions rather than low-level audio features. The model integrates user song interactions and chord sequences of music and uses an attention mechanism to differentiate the importance of different parts of the song. In this model, to make better use of the historical behavioral information of users, we refer to the design of the neural collaborative filtering algorithm to obtain embedding of users and songs. Under this basis, we designed a chord attention layer to mine users' fine-grained preferences for different parts of the music content. We conducted experiments with a subset of the last.fm-1b dataset. The experimental results demonstrate the effectiveness of the method proposed in this paper.**

***Index Terms*—Data Mining, Recommendation System, Music Information Retrieval, Machine Learning**

I. Introduction

In recent years, with the growth of the mobile internet, access to music from internet music platforms has become convenient. Exposure to new music productions through recommendation algorithms is becoming a new way of consuming music. As such, music recommendation algorithms are broadly applied in the industry and have captured the interests of many academics.

Common music recommendation algorithms can be broadly classified into two categories, one known as Collaborative Filtering (CF) and one known as Content-Based Filtering (CBF) [1]. The CF method learns the user's preferences from the user's interaction with the song. The CBF approach makes a recommendation based on similarities that are calculated using song's labels or audio contents. Since audio content reflects music's content directly, establishing a relationship between users and audio content will help achieve accurate music recommendations.

DOI reference number: 10.18293/SEKE2022-115

To this end, efforts have been made to hybridize the user's listening history with the audio content to generate recommendations. For example, Oord et al. introduce Convolutional Neural Network (CNN) into the music recommendation task [2]. They first obtain a representation vector of music items via the CF method and then learns the mapping of audio content to the music vector by training a CNN to generate an embedding vector for new music. Lee et al. propose a user embedding approach, which integrates user history records with audio content in the framework of Neural Collaborative Filtering (NCF) [3] and generates recommendations end-to-end [4].

Due to the success of audio features in tasks such as speech recognition, audio content is often represented as frequency domain features such as Mel-frequency cepstral coefficients or spectrograms in hybrid music recommendation algorithms [5]. However, while these low-level features have been shown to be suitable to for certain tasks, their discriminative power and semantics are limited. This makes them may be unsuitable for music classification, musical emotion recognition, or music recommendation tasks, which require better representations [6].

At the same time, users tend to have different degrees of preference for different segments of music content, termed fine-grained music preferences [7]. However, existing embedding methods for music content do not distinguish between different parts of the music at a fine-grained level. Instead, use CNNs or Recurrent Neural Networks (RNNs) to directly learn the mapping between the audio content and the embedding vector. Such coarse-grained embedding methods may trap existing methods into sub-optimal solutions. Therefore, how to appropriately represent music content in a hybrid music recommendation system and tap into the fine-grained music preferences of users for music content becomes a problem that needs to be addressed.

For the representation of music content, we propose to use higher-level music features such as the chord progression instead of audio features. In musical compositions, a chord progression is a continuous sequence of chords (e.g., C-G-Am-F) that describes the structure of music, which is the defining feature on which melody and rhythm are built. The chord features showed better performance than the low-level audio features in the music emotion classification task [8]. For the mining of users' fine-grained music preferences, we propose

to leverage attention mechanisms to learn users' preferences for different parts of the music, which have recently been used as an effective means to fine-grained data mining [9], [10].

We argue that users' fine-grained preferences on music content should be carefully mined in order to render better recommendations. To this end, we proposed a music recommendation model based on chord progression and attention mechanisms (COAT), which combines user-item interaction with music content. In the COAT model, we use a Generalized Matrix Factorization (GMF) layer to mimic matrix factorization and mining users' musical interests from their interaction with songs. Based on this, we design a chord attention layer to calculate user attention scores for the different chords on the chord progression and generate music content embedding. Finally, we use a prediction layer to combine the output of the GMF layer and the chord attention layer to predict the listening probabilities. We conducted experiments with a subset of the last.fm-1b dataset to assess the performance of our proposal. The experimental results show that our approach outperforms the baseline.

II. Related Work

A. Music Recommendation Algorithm

Deep learning-based music recommendation approaches usually use deep learning techniques to obtain a vector representation of a song from its audio content or metadata, known as an embedding vector. The obtained embedding vectors are then used to perform content-based recommendations, integrated into matrix factorization methods, or build hybrid music recommendation systems [5].

Oord et al. introduce deep learning techniques to music recommendation systems [2]. After obtaining the embedding vectors of the users and songs by implementing a matrix factorization method, they train a CNN to learn the mapping between the audio features and its embedding vector. This allows new generated music to obtain its embedding vector via this CNN without interaction with the users. Beyond the audio content, scholars try to integrate information in more modalities. Yi et al. propose a cross-modal variable autoencoder for content-based micro-video background music recommendations that integrates video content and audio content to form recommendations [11].

Since separating the process of acquiring music embedding vectors from acquiring user and song embedding vectors may produce sub-optimal solutions, some scholars consider an end-to-end manner to build hybrid recommendation systems [4]. Liang et al. suggest a hybrid approach. The method first learns the content features through a multi-layer neural network and subsequently integrates them into the matrix factorization as a prior [12]. Lee et al. suggest a deep content-user embedding model, which learns user and song embedding through a multi-layer neural network while using a CNN to learn the audio features of the song, and combines the two in an end-to-end way to finally generate recommendations [4]. Feng et al. proposed a hybrid music recommendation algorithm that combines user behavior and audio features to learn the fine-grained preferences of users for music content from multiple audio features by using an attention mechanism [7].

To sum up, much work has been done on integrating audio content into collaborative filtering recommender systems. However, these approaches have not yet explored the effects of higher-order music features with more explicit meanings in music recommendation tasks, nor have they been able to mine the fine-grained preferences of users for music content. The development of music information retrieval techniques and the application of attention mechanisms in recommender systems make it possible to fill this gap.

B. Attention-based Recommendation and Data Mining System

The human attention mechanism inspires the attention mechanism in deep learning. Like the attention to a specific part of the input in human vision, applying attention mechanisms in recommender systems allows the model to filter the most informative part from the input features. Hence reducing the influence of noisy data and thus improving the effectiveness of recommendations and bringing some interpretability [9], [13].

Wang et al. introduced the attention mechanism into the collaborative filtering method and proposed a dynamic user modeling approach based on the attention mechanism [14]. The method accurately portrays user interests by combining temporal information from calculating the degree of influence of the K items that the user has recently interacted with. The incorporation of the attention mechanism enhances the effectiveness of the CF method. Zhou et al. proposed a framework based on self-attention for modeling user behavior. The introduced self-attention mechanism demonstrated better performance and efficiency in their experiments than CNNs and RNNs [15].

Du et al. introduce an attention mechanism in the integration of user embedding vectors in a sarcasm detection task, which allows the features of various aspects of the user to be used effectively [16]. For the next point-of-interest recommendation task, Liu et al. proposed an attention-based category-aware GRU (ATCA-GRU) model [17]. The ATCA-GRU model can select the more significant parts of the relevant historical check-in trajectory to enhance the recommendation effect using the attention mechanism.

Gong et al. introduced an attention mechanism on the MOOC recommendation task. They fused meta-paths with contextual information by applying the attention mechanism on meta-paths of heterogeneous graphs to capture different students' different interests [18]. Shi et al. propose a method based on meta-paths and the attention mechanism [19]. Their proposed approach uses an attention mechanism to differentiate the importance of different meta-paths, which improves the effectiveness of recommendations and brings some interpretability.

In summary, attention mechanisms have been widely used with good results in various data mining tasks, which allows

us to apply them to the analysis of users' fine-grained music preferences.

III. Method

The architecture of our proposed COAT model is shown in Figure 1. COAT model takes the one-hot vector of users and songs and the corresponding audio file of the song as input, and the output is the probability that the user listens to the song. Within the model, we model the history of user-song interactions in a neural collaborative filtering framework designed by He et al [3]. Above this, we design a chord attention layer to differentiate the importance of different parts of the song. After obtaining the vectors representing the user's long-term interests and the vectors representing the music content, we use a stacked neural network (called a prediction layer) to learn the complex relationship between user behavior records and music content and thus generate recommendations.

A. Generalized Matrix Factorization Layer

The user's interaction history is the most important representation of the user's interests, and mining the user's interest preferences can help improve the recommendation effect. Matrix factorization is a representative technique for this task, and being able to mimic this technique in a recommendation model is the foundation for building a successful recommendation model [3].

For this, we use a generalized matrix factorization layer (GMF) to mimic the matrix factorization. This layer first receives one-hot vector representations of the user and the song, denoted by V_u^U and V_i^I. After the embedding operation is implemented, these high-dimensional one-hot vectors are mapped to lower-dimensional vectors, called user and song embedding vectors. In the work of He et al., after obtaining the embedding vectors for the user and the song, the probability of the user clicking on the song can be obtained directly from the inner vector product. Here we keep these two vectors and input them into the next part of the model. The function is as follow:

$$\mathrm{GMF}_{out} = W_u^T V_u^U \odot W_i^T V_i^I \tag{1}$$

where W_u and W_i are the learnable parameters that map V_u^U and V_i^I into embedding vectors, and $\odot$ denotes the element-wise product of vectors.

B. Chord Attention Layer

The chord attention layer first takes in the audio file and then performs chord extraction. At this stage, we use the chord-extractor tool[1] to extract the chords of the music. As the number of chords is often inconsistent from song to song, and neural networks cannot handle variable-length data, we uniformly padded the collected chord sequences to 100 by repeating them in order.It is worth mentioning that as chord progressions are always repeated in a song, thus this treatment will not affect the data quality.

[1]https://ohollo.github.io/chord-extractor/

After obtaining the chord sequence, we generate a random embedding vector representation for each chord. Each chord sequence can be represented by a matrix C, and C_i represents the i-th chord vector in the chord matrix. The value of i indicates the order in which the chord appear in the time dimension. And since different placements of the same chords produce different sounds (e.g., Am-F-C-G and C-G-Am-F are different chord progressions), we generate positional embedding for each position, representing as p_i.

Combining the user embedding vector e_u and the position vector p_i, we calculate the attention weight a_i of each chord embedding vector using the following equation:

$$a_i = \mathrm{h}^{\mathrm{T}} Relu\left(\mathrm{W}^T \begin{bmatrix} \mathrm{e_u} \\ \mathrm{C_i} \\ \mathrm{p_i} \end{bmatrix} + \mathrm{b}\right) \tag{2}$$

Where h, W and b are parameters, and $Relu$ is the Relu activation function.

After applying equation 2 to calculate the individual attention weights a_i, we normalized them using softmax with the following equation:

$$\beta_{\mathrm{i}} = \frac{\exp\left(\mathrm{a_i}\right)}{\sum_{\mathrm{j}=1}^{|\mathrm{C}|} \exp\left(\mathrm{a_j}\right)} \tag{3}$$

The normalized attention weights β_i represent the importance of the different parts of the song, with which we finally weighted and summed with the chord embedding vector to obtain the output of the chord attention layer Att_{out}. The function is as follow:

$$Att_{\mathrm{out}} = \sum_{i=1}^{|C|} \beta_i \cdot C_i \tag{4}$$

C. Prediction layer

After feeding the model with information on chord sequences that represent the content of the music, the preference relationship between the user and the song becomes complex. Therefore, the model needs to have a stronger fitting capability to fit this kind of preference relationship.

Thus, after obtaining the vector GMF_{out} which contains information on user behavior and the vector Att_{out} which contains information on song content from the other two parts of the model, we calculate the final listening probability using a stacked neural network with the following equation:

$$\hat{\mathrm{y}}_{\mathrm{ui}} = MLP\left(\begin{bmatrix} GMF_{\mathrm{out}} \\ Att_{\mathrm{out}} \end{bmatrix}\right) \tag{5}$$

MLP stands for the common multi-layer perceptron, whose number of layers and shape can be set flexibly. In this paper, we set its number of layers to 3 to avoid too many parameters causing overfitting. In terms of shape, we set it using a typical tower structure, where each layer has twice the number of neurons as the next layer. The premise of this approach is that setting smaller neurons in the higher-level neural network will enable more abstract information to be learned from the data [20].

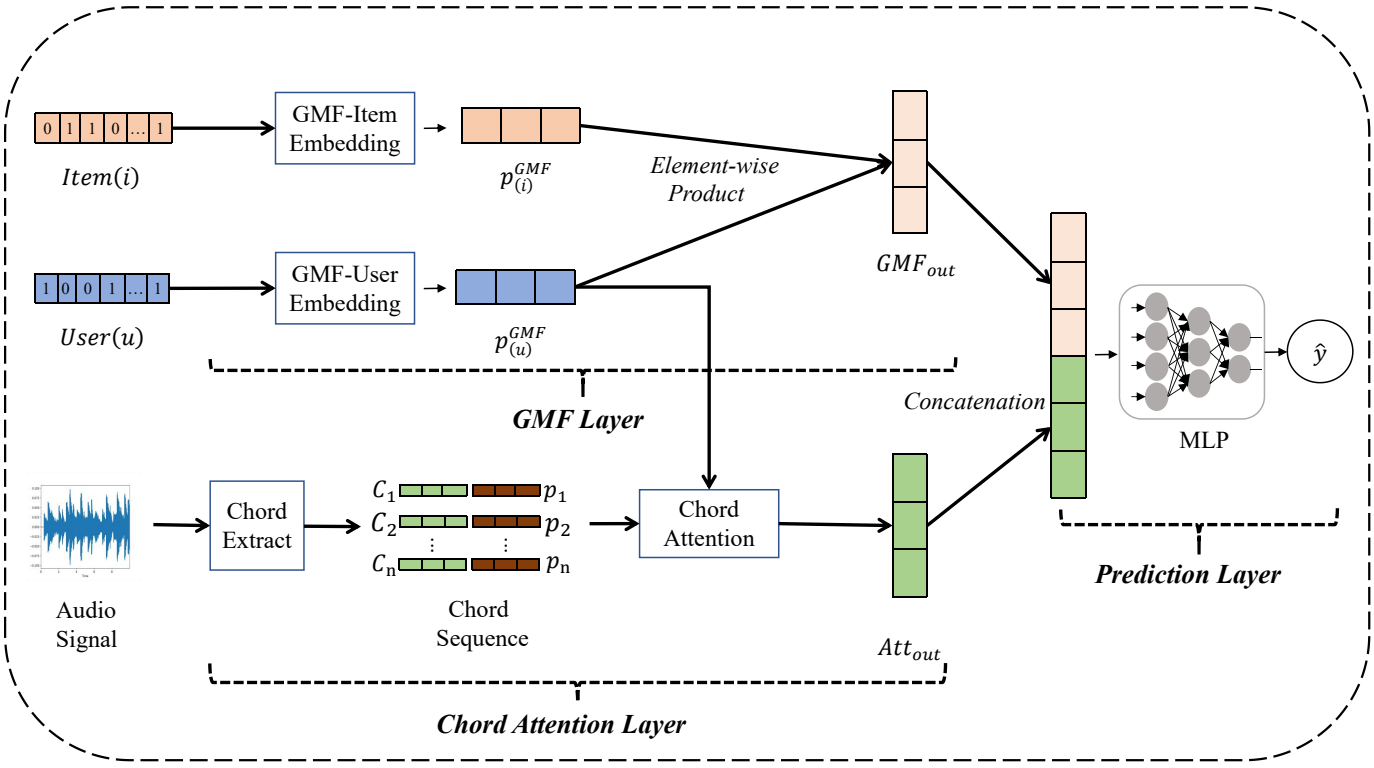


Fig. 1. The overall framework of our proposed COAT model

IV. Experiment Settings

In this section, we introduce the dataset used in the experiments. We then pose two research questions that we intend to answer in this paper to justify the proposed approach's effectiveness. Based on these questions, we design experiments and report their results.

A. Dataset Descriptions and Constructions

The experiments require a dataset containing the user's listening history and the song's audio file. For user listening history, we extracted a subset from the widely used last.fm-1b dataset [21]. As the dataset does not contain audio files of the songs, we downloaded the corresponding audio files from streaming platforms based on the collection of songs in the subset to form the dataset used in this paper.

Due to the size of the complete last.fm-1b dataset, conducting experiments on the complete data set would consume too much time. So we streamlined the last.fm-1b dataset in the following steps: Firstly, we used 2014 as the boundary to remove previous records from the complete dataset; Secondly, we filtered the top 10,000 popular songs from the song set to build a new subset; Finally, based on this subset, we removed listening records that were not relevant to it. We removed users with less than ten interaction records to ensure the dataset's quality. With 30,753 users, 10,000 songs and 1,533,245 interaction records, our dataset has a data sparsity of 99.50%.

B. Research Questions

- RQ1: Whether the chord attention mechanism improves the recommendation effect?
- RQ2: Whether the proposed method is better than traditional methods?

C. Experiment Design

To address the above research questions, we design four experiments accordingly.

- **Experiment 1:** To validate the effectiveness of our proposed attention mechanism, we conducted ablation experiments on it. As shown in Table I, we designed variants of the model with and without the attention mechanism (attention weights set to 1) and judged the effectiveness of the attention mechanism by comparing the performance of the recommendations.
- **Experiment 2:** We use a comparative experiment to verify whether our proposed model can obtain better results than the baseline approach. The chosen baseline methods include a traditional matrix factorization algorithm, a neural network-based collaborative filtering algorithm, and hybrid music recommendation algorithms based on audio features.

1) Parameter Settings.: The models involved in this paper use the same strategy for parameter settings. The range of searching for each hyperparameter is as follows: batch size is [128,256,512,1024], learning rate is [0.0001, 0.0005, 0.001, 0.005] and embedding size is [8,16,32,64].

2) Evaluation Protocols.: We used a strategy called leave-one-out to test the model's effectiveness, which has also been widely adopted in other work [22]. Regarding this strategy, the test set consists of one positive sample and several negative samples, where the positive sample is the last song in the user's listening record. For a given user, it would be time-consuming to add all songs in the dataset that have not been interacted with as negative samples to the test set for sorting, so we only sample 99 songs that have not been interacted with for a user as negative samples. This is a common strategy [23]. We use two common evaluation metrics to measure the effectiveness of ranking, *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain*(NDCG) [24]. The HR metric is given a 1 or 0 depending on whether the positive sample appears in the final top-n list. NDCG gives finer scores to positive samples based on where they appear in the top-n list. Higher scores are given to positive samples that appear higher up. We generate top-n recommendation lists for all users for each experiment round and use this to calculate two metrics, HR and NDCG. The average of all users' scores on both metrics is used as the final score of the model.

TABLE I
PERFORMANCE W.R.T. WITH OR WITHOUT ATTENTION

Embedding size	With Attention		W/o Attention	
	HR	NDCG	HR	NDCG
8	**0.581**	**0.344**	0.392	0.205
16	**0.629**	**0.390**	0.491	0.281
32	**0.647**	**0.421**	0.556	0.319
64	**0.653**	**0.442**	0.608	0.355

V. RESULTS AND DISCUSSIONS

A. Whether the chord attention mechanism improves the recommendation effect (RQ1)

Table I shows the results of the ablation experiments related to the attention mechanism proposed in this paper. As can be seen from the table, the model's performance is always better when attention is applied than when it is not, and there is a significant drop in performance when attention is not applied. This phenomenon suggests that the application of attention mechanisms to distinguish the importance of different parts from multimedia content in deep learning-based recommendation algorithms can positively impact the effectiveness of recommendations.

B. Performance Comparison (RQ2)

In this experiment, we selected a traditional matrix factorization approach, a deep learning-based collaborative filtering approach, and a hybrid audio content approach as baselines for comparison with the COAT model.

- **WMF** [25]: The method uses a weighted matrix factorization technique to obtain embedding vector of users and items and produces recommendations from the inner vector product.

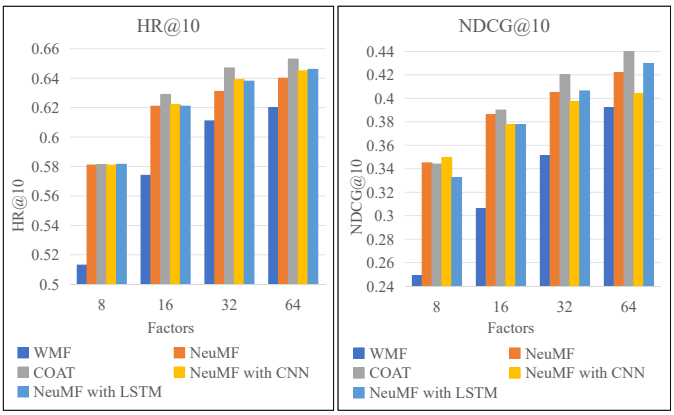


Fig. 2. Performance comparison of different methods.

- **NeuMF** [3]: The method uses deep neural networks to implement collaborative filtering and is the basis for many neural network-based recommendation algorithms.
- **NeuMF with CNN**: On top of NeuMF, CNN is used to process MFCCs features as a way to compare the performance of our proposed chord attention layer with that of the CNN-based approach.
- **NeuMF with LSTM**: On top of NeuMF, LSTM is used to process MFCCs features as a way to compare the performance of our proposed chord attention layer with that of the RNN-based approach.

Figure 2 shows the performance of each method under different predictive factors.

For the NeuMF method, the size of the predictive factor is the output dimension of the MLP and GMF layers in the method. For COAT, the size of the predictive factor is equal to the dimensions of the embedding vectors. For WMF, the number of predictive factors is equal to the embedding size. And for the LSTM-based approach and CNN-based approach, we use the Librosa [26] library to extract the MFCCs features from the audio to represent music content.

Figure 2 shows that the COAT model consistently achieves better results than the other methods on the two evaluation metrics when the predictor is greater than 8. NeuMF performs slightly better than the COAT model when the predictive factor size is 8. We consider this phenomenon because when the embedding dimension is too small, the COAT model has a smaller proportion of inputs related to user behavior, allowing the model to be influenced by the noise from the music content. This finding suggests that mining the user's behavioral history is crucial in designing recommendation algorithms. When the size of the neural network model is small, too much introduction may introduce more noise into the model and degrade the recommendation performance.

Compared with NeuMF, the LSTM-based approach can obtain some improvement, while the CNN-based approach will obtain more inferior results. This phenomenon indicates that audio features in matrix form, though similar to pictures, have more significant temporal features than local features. And because low-level audio features are complex, it is challenging

to process them properly in music recommendation models.

As shown in Figure 2. The gap between the COAT model and other methods becomes larger as the predictive factor increases. In our analysis, the input chord sequence information makes the preference relationship between the user and the song more complicated, which means that the recommendation model needs to have stronger fitting power to learn this preference relationship. As the predictors increase, the model's width increases, making the model a stronger fitting capability. This phenomenon demonstrates again that when applying deep learning techniques to recommender systems, the use of larger-scale models brings improved recommendation results.

VI. Conclusions and Future Work

In this paper, we propose using higher-order music features instead of lower-order frequency domain features to represent music content in the music recommendation model and differentiate the importance of music content properly. To this end, we propose a music recommendation model known as COAT, which is based on chord progression and attention mechanism. The model integrates user song interactions and chord sequences of music and uses an attention mechanism to differentiate the importance of different parts of the song. The experimental results demonstrate the effectiveness of the method proposed in this paper.

In the future, we will explore introducing more higher-order music features, such as melody, rhythm, lyrics, etc., into the recommendation model based on the latest advancement in music information retrieval. At the same time, we will consider integrating this work with the existing graph neural network-based collaborative filtering recommendation model to achieve better recommendation results.

Acknowledgement

This work is partially supported by the Project of Beijing Municipal Education Commission (No.KM202110005025), the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions (CIT&TCD20190308), and Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education.

References

[1] M. Schedl, H. Zamani, C. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 95–116, 2018.

[2] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in neural information processing systems*, 2013, pp. 2643–2651.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173–182.

[4] J. Lee, K. Lee, J. Park, J. Park, and J. Nam, "Deep content-user embedding model for music recommendation," *arXiv: Information Retrieval*, 2018.

[5] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019.

[6] P. Knees and M. Schedl, *Music similarity and retrieval: an introduction to audio-and web-based strategies*. Springer, 2016, vol. 9.

[7] W. Feng, T. Li, H. Yu, and Z. Yang, "A hybrid music recommendation algorithm based on attention mechanism," in *International Conference on Multimedia Modeling*. Springer, 2021, pp. 328–339.

[8] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, I.-B. Liao, and H. H. Chen, "Automatic chord recognition for music classification and retrieval," in *2008 IEEE International Conference on Multimedia and Expo*. IEEE, 2008, pp. 1505–1508.

[9] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 335–344.

[10] Y. Yu, S. Luo, S. Liu, H. Qiao, Y. Liu, and L. Feng, "Deep attention based music genre classification," *Neurocomputing*, vol. 372, pp. 84–91, 2020.

[11] J. Yi, Y. Zhu, J. Xie, and Z. Chen, "Cross-modal variational auto-encoder for content-based micro-video background music recommendation," *IEEE Transactions on Multimedia*, 2021.

[12] D. Liang, M. Zhan, and D. P. Ellis, "Content-aware collaborative music recommendation using pre-trained neural networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*. International Society for Music Information Retrieval, 2015, pp. 295–301.

[13] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, p. 5, 2019.

[14] R. Wang, Z. Wu, J. Lou, and Y. Jiang, "Attention-based dynamic user modeling and deep collaborative filtering recommendation," *Expert Systems with Applications*, vol. 188, p. 116036, 2022.

[15] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018, pp. 4564–4571.

[16] Y. Du, T. Li, M. S. Pathan, H. K. Teklehaimanot, and Z. Yang, "An effective sarcasm detection approach based on sentimental context and individual expression habits," *Cognitive Computation*, pp. 1–13, 2021.

[17] Y. Liu, A. Pei, F. Wang, Y. Yang, X. Zhang, H. Wang, H. Dai, L. Qi, and R. Ma, "An attention-based category-aware gru model for the next poi recommendation," *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3174–3189, 2021.

[18] J. Gong, S. Wang, J. Wang, W. Feng, H. Peng, J. Tang, and P. S. Yu, "Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 79–88.

[19] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top- n recommendation with a neural co-attention model," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] M. Schedl, "The lfm-1b dataset for music retrieval and recommendation," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, 2016, pp. 103–110.

[22] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1341–1350.

[23] A. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 278–288.

[24] X. He, T. Chen, M. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1661–1670.

[25] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Eighth IEEE International Conference on Data Mining*, 2009, pp. 263–272.

[26] B. Mcfee, C. Raffel, D. Liang, D. Ellis, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science Conference*, 2015.

Deep Correlation based Concept Recommendation for MOOCs

1st Shengyu Mao
Zhejiang University of Technology
Hangzhou, China
201806062515@zjut.edu.cn

2nd Pengyi Hao
Zhejiang University of Technology
Hangzhou, China
haopy@zjut.edu.cn

3th Cong Bai
Zhejiang University of Technology
Hangzhou, China
congbai@zjut.edu.cn

***Abstract*—The current course recommendation in massive open online courses (MOOCs) usually ignores students' interests in some certain type of knowledge concepts, resulting in low completion of most courses. Therefore, it requires a concept recommendation to help students accurately choose courses in MOOCs. In this paper, we propose Deep Correlation based Concept Recommendation (DCCR) for MOOCs. It gathers the interactive information obtained by different entities through meta-paths in MOOCs and extracts the semantic information of concepts. To deeply capture the correlation information among users, a multi-relation graph is built to generate the correlation features which aggregates the abundant information under different meta-paths. Then through the graph convolutional neural networks, entity embeddings of users and knowledge concepts are generated. Additionally, a concatenation-based fusion function is designed to get the final joint representations reasonably. By verifying on two public datasets, experiments show that DCCR outperforms the state-of-the-art methods.**

***Index Terms*—concept recommendation, correlation feature, concatenation based fusion, moocs**

I. Introduction

MOOCs have been developing rapidly in recent years, providing users with a convenient way of education [12]. The emergence of MOOCs' platforms has completely changed the entire education field. However, according to statistics, there is a very low completion rate of online courses [10]. So there are lots of works concentrating on course recommendation in MOOCs. But barely focus on courses would lead to some problems. 1) The normal course recommendation could probably cause students to take courses that they are not attracted to. For example, some computer vision courses only cover knowledge about geometric, and others may cover deep learning, which could mislead the result in recommendation. 2) The content and focus of similar courses are different. For instance, in advanced mathematics, some courses are based on geometry, and some courses are based on calculus, which is difficult for students to select [16]. 3) Students with different prerequisite background may require totally different [5], but if recommending in the traditional way, students may not realize that they have not learnt some necessary prerequisite knowledge until taking certain courses [8]. Therefore, the MOOCs need to accurately locate the learning needs of students, which makes it necessary to do concept recommendation, a more fine-grained recommendation task.

The existing course or concept recommendations are mainly divided into two categories. One is the methods based on collaborative filtering (CF), which considers the historical interaction behavior of users with other online resources (e.g., videos, courses), and explores the potential of the same preference [9]. For example, He et al. proposed a neural network model to evaluate the similarity between items by using an attention network [7]. Elbadrawy et al. used neighborhood-based user collaborative filtering to design a course ranking model [6]. These methods have achieved great success in recommendation courses, but suffering from the problem of data sparsity and cold start, so the performance is limited. The other one constructs the MOOCs information into heterogeneous information networks, and utilize meta-paths to guide the dissemination of student preferences. It always captures the corresponding fruitful semantic relationships between different types of entities, learns the embedding, and finally generates a recommendation list through matrix factorization [16]. Vashishth et al. [14] proposed a three-way neural interaction model to use the rich meta-path-based information for recommendation. Gong et al. [16] proposed a method to capture the representation of different types of entities in heterogeneous information networks, fused the content features of entities to do recommendation. However, due to the relative independence of different meta-path relationships, this kind of methods can not completely capture the interactive information from the heterogeneous networks in MOOCs, and may lose some inner information among different meta-paths, resulting in an unsatisfactory performance.

In this paper, we excavate the deep correlation of users. Motivated by some efficient multi-relational graph methods like [15], we propose deep correlation based concept recommendation (DCCR) for MOOCs, where a multi-relational graph is constructed to deeply capture the inner correlation of users among different meta-paths. With the deep correlation feature extracting from the multi-relational graph , representations for users in MOOCs can be better learnt from graph convolutional network to accurately reflect users' preference. Meanwhile, we deeply extract the information contained in concepts themselves including names, definitions, etc. as auxiliary features, which makes the recommendation system

DOI reference number: 10.18293/SEKE2022-107

more effective. We also propose a concatenation-based fusion function to better combine the entity representations under different meta-paths, and finally get the rating matrix from users towards knowledge concepts. The key contributions of this paper can be summarized as: (i) deep correlation features of users are generated by constructing a multi-relational graph among different meta-paths, which leads to better embedded representations of user entities. (ii) a feature fusion function is proposed by concentrating on the different entity embeddings under different meta-paths, which leads to more reasonable representation for users and concepts. (iii) the DCCR is evaluated on two publicly available real datasets MOOCCube [17] and XuetangX [16], not only the performance in each dataset is compared, but also the difference between the two datasets are analysised.

II. Proposed Method

A. Probelm Statement

In the recommendation task, MOOCs' data usually includes five specific entities (user(**u**), knowledge concept(**k**), video(**v**), course(**c**), teacher(**t**) [13], [16]). Additionally, there are abundant text information along with knowledge concepts, including their definitions, descriptions, and classifications. The purpose is to generate concept recommendation list for each user. The framework of DCCR is shown in Fig. 1, and the explanations of notations are given in Table I.

TABLE I: Notations and explanations.

Notation	Explanation
S_k	semantic feature of knowledge concepts
C_u	correlation feature of users
d_u, d_k	the dimension of correlation feature and semantic feature
N_u, N_k	the number of users and knowledge concepts
MP^u, MP^k	the meta-path sets of users and knowledge concepts
A_{mp^u}, A_{mp^k}	the adjacency matrix sets of users and knowledge concepts
G	the correlation triad set
z	the correlation triad threshold
f_{mp^u}, f_{mp^k}	the representation sets of users and knowledge concepts
p	the scale parameter
F_k, F_u	the final representations of users and knowledge concepts
d_F	the dimension of the final representations of users and knowledge concepts
$r_{u,k}$	the true rating of user u to knowledge concept k
$\hat{r_{u,k}}$	the predicted rating of user u to knowledge concept k
x_u	the latent factors of user u
y_k	the latent factors of knowledge concept k
t_u, t_k	the parameters to integrate the F_u and F_k in the same space
β_u, β_k	the tuning parameters
δ	the regularization parameters

B. Semantic Feature Extraction

Ordinarily, the name of a concept, almost the generalization of itself, contains rich semantic information. Moreover, just like the name, there is also a wealth of information in concept's subject classifications, e.g., the subject classifications of 'Binary tree' are 'theoretical computer science' and 'data structure', so at least two specific subject classifications can be collected for this knowledge concept.

After separating the text informations into words, a parameter c is set as the number of classification we chose, and then the word vectors $S_{name}, S_{class_1}, S_{class_2}, \cdots, S_{class_c} \in \mathbb{R}^{N_k \times d_v}$ are generated by *Word2Vec* [18], in which N_k indicates the total number of knowledge concepts, and d_v indicates the dimension of word vectors. We stitch the word vectors together to get the semantic feature of concepts $S_k \in \mathbb{R}^{N_k \times d_k}$, $S_k = [S_{name}, S_{class_1}, S_{class_2}, \cdots, S_{class_c}]$, here $d_k = d_v \times (c+1)$ indicates the dimension of semantic feature.

C. Meta-Path Adjacency Matrices

Given the interactive information between different kinds of entities in MOOCs (e.g., user *u3125* have learned courses *c254* and *c617*), we build the interactive matrices between entities, including user-click-knowledge concept matrix, user-watch-video matrix, user-learn-course matrix, user-learn-course-taught by-teacher matrix, knowledge concept-included by-video matrix and knowledge concept-involved-course matrix. Each element in each matrix belongs to $\{0, 1\}$, which represents the interaction between two specific entities.

Meta-path [4] means the semantic path that connects different entities and illustrates the relational information in the dataset. Here, meta-paths are defined like $\mathbf{u} \xrightarrow{watch} \mathbf{v} \xrightarrow{watch^{-1}} \mathbf{u}$, which indicates that two different users are connected because they have watched the same video; $\mathbf{k} \xrightarrow{clicked\ by} \mathbf{u} \xrightarrow{clicked\ by^{-1}} \mathbf{k}$ means that two different knowledge concepts have been clicked by the same user. In this way, the meta-path sets $MP^u = \{mp_1^u, mp_2^u, \cdots, mp_m^u\}$ and $MP^k = \{mp_1^k, mp_2^k, \cdots, mp_n^k\}$ are set for users and knowledge concepts respectively, where m, n indicate the number of meta-paths of users and knowledge concepts.

With the interactive information matrices and the meta-path sets, the corresponding adjacency matrix sets $A_{mp^u} = \{A_{mp_1^u}, A_{mp_2^u}, \cdots, A_{mp_m^u}\}$ and $A_{mp^k} = \{A_{mp_1^k}, A_{mp_2^k}, \cdots, A_{mp_n^k}\}$ are generated for users and concepts. Taking users as an example, A_{mp^u} is generated and normalized by the following formula,

$$A_{mp_i^u} = \mathit{Norm}(L_u^e \cdot L_u^{eT}), \tag{1}$$

here L_u^e is the corresponding interactive matrix of mp_i^u(e.g., the user-watch-video matrix is the corresponding interactive matrix of meta-path $\mathbf{u} \xrightarrow{watch} \mathbf{v} \xrightarrow{watch^{-1}} \mathbf{u}$), $A_{mp_i^u}$ is the normalized adjacency matrix. After iterating through all the meta-paths in MP^u, the meta-path adjacency matrix A_{mp^u} of users is obtained. In the same way, the meta-path adjacency matrix of knowledge concepts $A_{mp^k} = \{A_{mp_1^k}, A_{mp_2^k}, \cdots, A_{mp_n^k}\}$ is also generated.

D. Correlation Feature Extraction

The meta-path adjacency matrix sets generated above have captured many significant information from MOOCs. However, the relative independence of those adjacency matrices loses lots of correlation information among meta-paths. Fig. 2 shows the contrast between with and without correlation among meta-paths. Since there are various connections between users and other entities, the correlation information among different users' preferences can be deeply gathered.

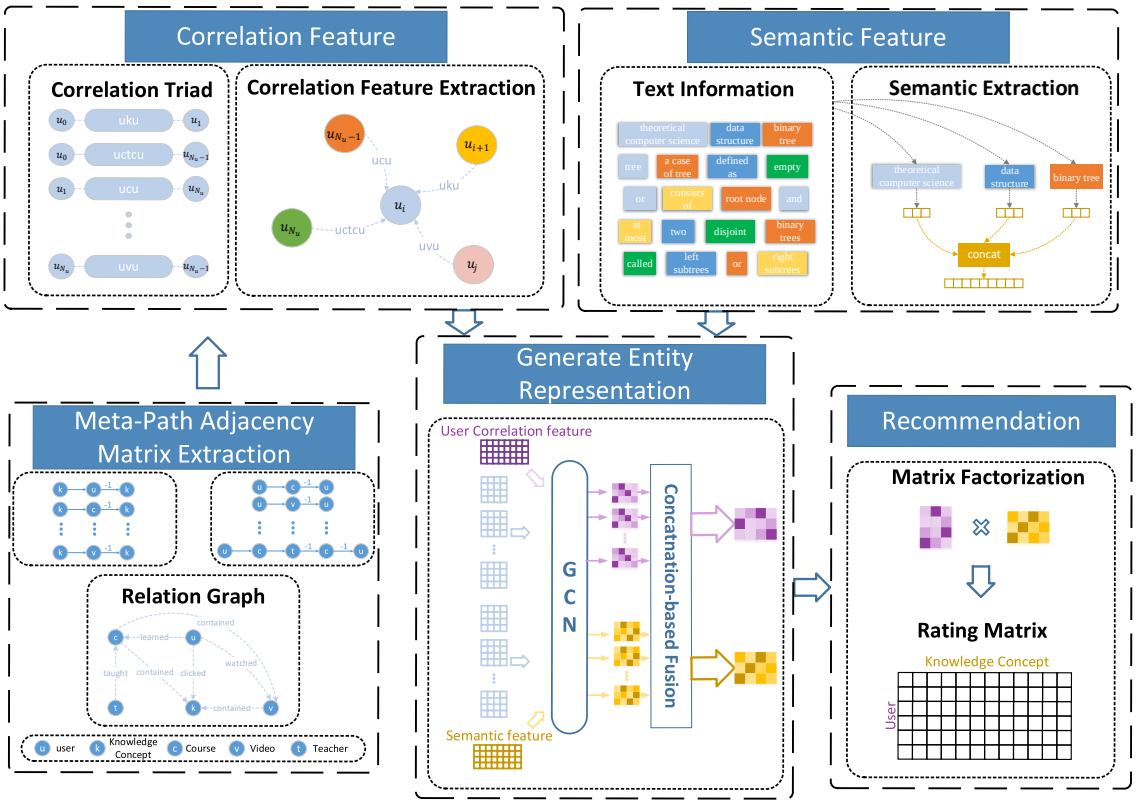


Fig. 1: The framework of DCCR for MOOCs.

First, a threshold z is used to select the users that are strongly correlated. Then, a user correlation triad set G is created. For specific user entities u_i and u_j, if their corresponding value in $A_{mp_q^u}(q \in [1, m])$ is lager than z, which means that these two users are correlated in mp_q^u, then the correlation information (u_i, r, u_j) will be written into G, where r is the meta-path mp_q^u.

Based on the triad set G, we build a multi-relational graph by $Graph = (\mathcal{V}, \mathcal{R}, \mathcal{E}, \mathcal{X}, \mathcal{Z})$, where $\mathcal{V}$ is the node set of entities, $\mathcal{R}$ is the correlation set of entities, $\mathcal{E}$ is the enlarged correlation triad set, $\mathcal{X}$ and $\mathcal{Z}$ are the initial feature of nodes and relations, $\mathcal{V}, \mathcal{R}, \mathcal{E}$ are built by G. For every triad $(u_i, r, u_j) \in G$, u_i, u_j are involved in $\mathcal{V}$; $\mathcal{R} = \mathcal{R}' \cup \mathcal{R}'_{inv} \cup \{Se\}$, where $\mathcal{R}' = \{r | (u_i, r, u_j) \in G\}$, $\mathcal{R}'_{inv} = \{r^{-1} | (u_i, r, u_j) \in G\}$, Se means the self-loop correlation; $\mathcal{E} = \{(u_i, r, u_j) | (u_i, r, u_j) \in G\} \cup \{(u_j, r^{-1}, u_i) | (u_i, r, u_j) \in G\} \cup \{(u, Se, u) | u \in \mathcal{V}\}$. After that, we get the embeddings with following rules,

$$h^1_{u_j} = tanh(\sum_{(u_i, r) \in N(u_j)} W^1_{\lambda(r)} \phi(x_{u_i}, z_r)), \tag{2}$$

$$h^2_{u_j} = tanh(\sum_{(u_i, r) \in N(u_j)} W^2_{\lambda(r)} \phi(h^1_{u_i}, h^1_r)), \tag{3}$$

where $N(u_j)$ is a set of immediate neighbours of u_j for its outgoing edges, $\phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a composition operator, $\lambda(r)$ indicates the relation type of r, $W^k_{\lambda(r)}$ is a relation-type shared parameter at k-th layer, x_{u_i} and z_r are the initial features of node u_i and relation r, $h^1_{u_j}$ is the feature of node u_j generated at the first layer, h^1_r is the representation of relation generate at first layer which follows the rule $h^{k+1}_r = W^k_{rel} h^k_r$, here W^k_{rel} are the shared parameters for each relation. $h^2_{u_j} \in \mathbb{R}^{d_u}$ is the output of the second layer as well as the correlation feature of u_j. All the parameters are randomly initialized. Finally the correlation feature matrix $C_u \in \mathbb{R}^{N_u \times d_u}$ is generated as,

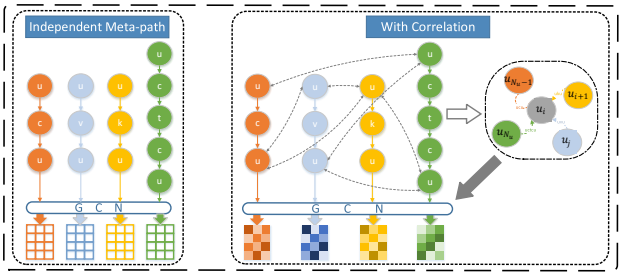


Fig. 2: Contrast between with and without correlation

$$C_u = [h^2_{u_1}, h^2_{u_2}, \cdots, h^2_{u_{N_u}}]^T, \tag{4}$$

here, N_u is the number of users and d_u is the dimension of correlation feature.

E. Concatenation-based Representations Learning

Unlike the specific values in meta-path adjacency matrices A_{mp}, the triads in the multi-relation graph can not tell the degree that two users are correlated. So it's not efficient to

take the correlation feature C_u as the final representations of users. We need to further generate the representations of users and concepts under every meta-path. Given the semantic feature S_k, correlation feature C_u, and the meta-path adjacency matrices A_{mp^k} and A_{mp^u} as inputs, graph convolutional network (GCN) [19] is adopted with a layer-wise propagation rule for both users and concepts. Taking user as an example, the propagation layer is defined as

$$h^{(l+1)}_{mp^u_i} = ReLU(P_{mp^u_i} h^l_{mp^u_i} W^l). \tag{5}$$

Here $h^{(l+1)}_{mp^u_i}$ indicates the new representation of users under mp^u_i at layer $l+1$. $P_{mp^u_i} = \widetilde{D}^{-\frac{1}{2}} \cdot (A_{mp^u_i} + I) \cdot \widetilde{D}^{-\frac{1}{2}}$,$\widetilde{D} = diag((A_{mp^u_i} + I) \cdot \mathbf{1})$, I is the identity matrix, $\mathbf{1}$ is the all-ones vector, W^l is the shared trainable weight matrix at layer l for every meta-path. Particularly, we take C_u as $h^0_{mp^u_i}$ for users at the first layers, and take the output at the third layer as the representation under mp^u_i,

$$f_{mp^u_i} = h^3_{mp^u_i}. \tag{6}$$

After iterating through all the matrices in $A_{mp^u_i}$, we ultimately get the representation sets $f_{mp^u} = \{f_{mp^u_1}, f_{mp^u_2}, \cdots, f_{mp^u_m}\}$ for users. In the same way, we take S_k as $h^0_{mp^k_i}$ for knowledge concept at the first layer to adopt GCN and generate the representations as

$$f_{mp^k_i} = h^3_{mp^k_i}. \tag{7}$$

Finally, $f_{mp^k} = \{f_{mp^k_1}, f_{mp^k_2}, \cdots, f_{mp^k_m}\}$ is abtained for concepts.

In order to evenly consider meta-paths, joint representation by fusing representations under different meta-paths should be considered. However, in MOOCs, users' interactivity under different meta-paths are quite dissimilar. To solve this problem, we design a concatenation-based fusion function, which generates fusion weights by concentrating on representations under both the current meta-path and the others at the same time. In this way, we obtain a reasonable joint representation, which can reflects the association between meta-paths effectively. Taking users as an example, fusion weight $\alpha_{mp^u_i}$ of mp^u_i is calculated as

$$\alpha_{mp^u_i} = softmax(v(tanh(w_1 f_{mp^u_i} p + w_2 \mathtt{f}_{mp^u_i}(1-p) + b))), \tag{8}$$

where $\mathtt{f}_{mp^u_i} = \dfrac{1}{m-1} \cdot \sum_{j \neq i}^{m} f_{mp^u_j}$ is the second concerned object of mp^u_i, v, w_1, w_2, b are trainable parameters, $p \in (0,1)$ is a scale hyper-parameter. Then the final joint representation of user $F_u \in \mathbb{R}^{N_u \times d_F}$ is calculated as

$$F_u = \sum_{i=1}^{m} \alpha_{mp^u_i} f_{mp^u_i}. \tag{9}$$

Similarly, the final representation of knowledge concept $F_k \in \mathbb{R}^{N_k \times d_F}$ is calculated as

$$F_k = \sum_{i=1}^{m} \alpha_{mp^k_i} f_{mp^k_i}. \tag{10}$$

Algorithm 1 shows how to generate F_u and F_k.

Algorithm 1: Generate the representations of users and concepts

Input: The interactive information between different entities in MOOCs, the meta-path sets of users and concepts MP^u, MP^k, the text information of concepts
Output: The representations F_u, F_k of users and concepts

Initialize $A_{mp^u}, A_{mp^k}, f_{mp^u}, f_{mp^k}$ as empty lists
Extract semantic feature S_k from text information
for *each* $mp^k_i \in MP^k$ **do**
  Calculate $A_{mp^k_i}$ by Eq (1)
  Add $A_{mp^k_i}$ to A_{mp^k}
for *each* $mp^u_i \in MP^u$ **do**
  Calculate $A_{mp^u_i}$ by Eq (1)
  Add $A_{mp^u_i}$ to A_{mp^u}
Represent a multi-relational graph by A_{mp^u}
Extract correlation feature C_u by Eq (2,3,4)
for *each* $A_{mp^u_i} \in A_{mp^u}$ **do**
  $h^0_{mp^u_i} \leftarrow C_u$
  for $l = 0$ *to* 2 **do**
    Calculate $h^{l+1}_{mp^u_i}$ by Eq (5)
  $f_{mp^u_i} = h^3_{mp^u_i}$ according to Eq (6)
  Add $f_{mp^u_i}$ to f_{mp^u}
for *each* $A_{mp^k_i} \in A_{mp^k}$ **do**
  $h^0_{mp^k_i} \leftarrow S_k$
  for $l = 0$ *to* 2 **do**
    Calculate $h^{l+1}_{mp^u_i}$ by Eq (5)
  $f_{mp^k_i} = h^3_{mp^k_i}$ by Eq (7)
  Add $f_{mp^k_i}$ to f_{mp^k}
for *each* $f_{mp^u_i} \in f_{mp^u}$ **do**
  Calculate $\alpha_{mp^u_i}$ by Eq (8)
Generate F_u by Eq (9)
for *each* $f_{mp^k_i} \in f_{mp^k}$ **do**
  Calculate $\alpha_{mp^k_i}$ by Eq (8)
Generate F_k by Eq (10)
return F_u, F_k

F. Concept Recommendation for User

Lastly, an extend matrix factorization is utilized to complete the recommendation task. The predicted rating matrix $\hat{r_{u,k}}$ is got as follows,

$$\hat{r_{u,k}} = x_u^T y_k + \beta_u F_u^T t_k + \beta_k t_u^T F_k, \tag{11}$$

where x_u and y_k are randomly initialized latent factors of user and knowledge concept, t_u and t_k are parameters that make sure F_u and F_k to be in the same space, β_u and β_k are tuning parameters. We define the following loss function for reaching an appropriate rating prediction,

$$\begin{aligned} Loss = & \frac{1}{N_u \times N_k} \sum_{u=1}^{N_u} \sum_{k=1}^{N_k} (r_{u,k} - \hat{r_{u,k}})^2 + \\ & \delta(||x_u||_2 + ||y_k||_2 + ||t_u||_2 + ||t_k||_2), \end{aligned} \tag{12}$$

where $r_{u,k}$ is the target rating matrix of user on knowledge concept, δ is the regularization parameter. Finally, with the rating matrix of user on knowledge concept, the concepts are recommended with the highest rating for each user.

III. Experiment

A. Datasets

To evaluate the effectiveness of the proposed method, we adopt two datasets, MOOCCube [17] and the real data from XuetangX [16]. MOOCCube [17] is a large-scale data repository of over 700 MOOC courses, 100k concepts, 8 million student behaviors with an external resource. The abundant data of MOOCCube was mostly obtained from Baidubaike, Wikipedia, and Termonline. We divide the interactive behaviors of users to concepts into training set and test set with a

ratio of 8:2. XuetangX [16] includes a training set occurring between October 1st, 2016 and December 30th, 2016 and a test set with the data occurring between January 1st, 2018 and March 31st, 2018. It contains 7,020 MOOC courses 43,405 videos, 1,029 course concepts, and 9,986 real MOOC users. For both datasets, we paired 99 randomly sampled negative instances with 1 positive instance for each users, and output the prediction rating [7].

B. Evaluation Metrics and Implementation Details

Several metrics are utilized to evaluate the recommendation methods. $HR_{@K}$ is a common recall measure that shows the percentage of top-K recommendations that were successful. $NDCG_{@K}$ [3] is used to evaluate the differences between this ranking list and the user's actual interaction list. MRR [7] is used for evaluating any process that produces a list of possible responses to a sample of queries. Additionally, AUC is also used as a metric.

The methods are run in the environment of python-3.7, tensorflow-1.13.1. When extracting correlation features, We set the initial dimension size for nodes and relations to 100 and the output dimension to 200. When learning representations, we set the dimension to 256, 128 and 64 at the first, second and output layer, respectively. Moreover, we set the dropout rate to be 0.5 and the latent dimension to be 30 in MF. As for the learning rate, we set it to be 0.001, and implement an exponential learning rate decays every 100 steps.

C. Analysis of the Proposed Method

1) Evaluation of Meta-path Combinations: We emphatically analyse the influence of the selection and combination of different meta-paths through the whole recommendation task by referring the combinations in [16], but we further conduct a detailed analysis in two different datasets. Specifically, we consider the following meta-path combinations in both datasets, including $mp_1 : \mathbf{u} \rightarrow \mathbf{k} \stackrel{-1}{\rightarrow} \mathbf{u}, mp_2 : \mathbf{u} \rightarrow \mathbf{c} \stackrel{-1}{\rightarrow} \mathbf{u}, mp_3 : \mathbf{u} \rightarrow \mathbf{v} \stackrel{-1}{\rightarrow} \mathbf{u}$ and $mp_4 : \mathbf{u} \rightarrow \mathbf{c} \rightarrow \mathbf{t} \stackrel{-1}{\rightarrow} \mathbf{c} \stackrel{-1}{\rightarrow} \mathbf{u}$.

From Table II, the rank of effectiveness is $mp_1 > mp_3 > mp_4 > mp2$ in MOOCCube, and $mp_3 > mp_1 > mp2 > mp_4$ in XuetangX. Additionally, the combinations of meta-paths perform better than the individual, and the tendency of the effect is the same as the individual. For instance, in MOOCCube the performance of $mp_{1\&3}$ is better than $mp_{1\&2}$, and in XuetangX the performance of $mp_{1\&2\&3}$ is better than $mp_{1\&2\&4}$, which indicates that users have dissimilar interactive behaviors under different meta-paths. In general, it works best when combining all the four meta-paths. In MOOCCube, the AUC of $mp_{1\&2\&3\&4}$ is 4.17%, 6.18%, 4.78%, 5.67% higher than mp_1, mp_2, mp_3, mp_4, respectively. In XuetangX, the AUC of $mp_{1\&2\&3\&4}$ is 6.74%, 9.79%, 6.48%, 10.07% higher than mp_1, mp_2, mp_3, mp_4, respectively.

From Table II, it exhibits a larger increase in XuetangX than in MOOCCube when combing more meta-paths. For example, the AUC grows 0.75% from $mp_{1\&2}$ to $mp_{1\&2\&3}$ in MOOCCube, while it grows 2.01% in XuetangX. It is because of that XuetangX has more courses, videos and teacher entities besides users and concepts, depending on which we can extract more complete correlation features from XuetangX.

TABLE II: Results of different meta-path combinations

meta-path	MOOCCube				XuetangX			
	$HR_{@5}$	$NDCG_{@5}$	MRR	AUC	$HR_{@5}$	$NDCG_{@5}$	MRR	AUC
mp_1	0.6336	0.5184	0.5247	0.9077	0.5871	0.4166	0.3933	0.8909
mp_2	0.6027	0.5013	0.4735	0.8876	0.4559	0.3184	0.3115	0.8604
mp_3	0.6292	0.5125	0.5179	0.9016	0.6058	0.4218	0.3954	0.8937
mp_4	0.6125	0.5087	0.4893	0.8927	0.4456	0.3123	0.3072	0.8576
$mp_{1\&2}$	0.6748	0.5689	0.5564	0.9194	0.5655	0.3919	0.3699	0.9058
$mp_{1\&3}$	0.6984	0.6134	0.6083	0.9236	0.5969	0.4423	0.4353	0.9185
$mp_{1\&4}$	0.6851	0.5868	0.5732	0.9204	0.5828	0.4025	0.3795	0.9077
$mp_{2\&3}$	0.6927	0.6059	0.5913	0.9227	0.6502	0.4723	0.4439	0.9161
$mp_{2\&4}$	0.6624	0.6106	0.5718	0.9176	0.4934	0.3466	0.3353	0.8586
$mp_{3\&4}$	0.6858	0.6048	0.5883	0.9208	0.5257	0.3697	0.3575	0.8993
$mp_{1\&2\&3}$	0.7279	0.6364	0.6360	0.9311	0.7162	0.5464	0.5168	0.9386
$mp_{1\&2\&4}$	0.7167	0.6222	0.6207	0.9246	0.6727	0.5285	0.4673	0.9199
$mp_{1\&3\&4}$	0.7214	0.6347	0.6301	0.9302	0.7106	0.5463	0.5191	0.9367
$mp_{2\&3\&4}$	0.7125	0.6208	0.6264	0.9274	0.6924	0.5307	0.4916	0.9213
$mp_{1\&2\&3\&4}$	**0.7542**	**0.6708**	**0.6637**	**0.9494**	**0.7851**	**0.6118**	**0.5766**	**0.9583**

TABLE III: Results of different meta-path combinations

Method	MOOCCube				XuetangX			
	$HR_{@5}$	$NDCG_{@5}$	MRR	AUC	$HR_{@5}$	$NDCG_{@5}$	MRR	AUC
Ave-Fusion	0.6338	0.5387	0.5292	0.8852	0.3923	0.2602	0.2602	0.8506
Loc-Fusion	0.7364	0.6504	0.6426	0.9414	0.7247	0.5552	0.5204	0.9475
Con-Fusion	**0.7542**	**0.6708**	**0.6637**	**0.9494**	**0.7851**	**0.5833**	**0.5491**	**0.9583**

2) Comparison of Different Fusion Functions: In order to verify the efficiency of the proposed fusion function, we evaluate the recommendation task when using different fusion functions in the two datasets, including *Ave*-Fusion that takes an average of each vector, *Loc*-Fusion that is a location-based fusion [11], and *Con*-Fusion is a concatenation-based fusion designed in our method. Table III turns out that in both datasets, the effect of *Ave*-Fusion is much worse than the others. *Con*-Fusion works the best, the AUC of *Con*-Fusion is 0.8% higher than *Loc*-Fusion in MOOCCube, and 1.08% higher than *Loc*-Fusion in XuetangX, which means that concatenation-based fusion designed in our method can gather more associated information from different meta-paths.

Fig. 3: The performance of different z in two datasets

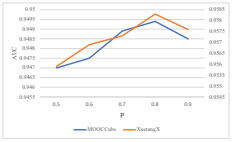


Fig. 4: The performance of different p in two datasets

3) Evaluation of Model Parameters: In DCCR, the triad threshold z is an important parameter. Fig. 3 shows how z effect the result. The best results in the two datasets are obtained when z is equal to 0.38 and 0.42, respectively. When z is too small, too much useless correlation information may be got, which will affect the performance of the method. While if z is too large, there will be a lack of correlation information leading to a bad performance.

In addition, in the proposed fusion function, we consider the impact of different values of scale parameter $p \in [0, 1]$, Fig. 4 shows the results. When $p = 0.8$, the best results are obtained in both datasets. If p is too small, the effect will be reduced due to insufficient attention to the representation itself. If too large, it will be hard to fuse the association with the other meta-paths.

D. Comparison with Other Methods

We compare with the following methods. *MLP* [2] that applies a multi-layer perceptron to user representations and the target knowledge concept representations, *FISM* [1] that is an item-to-item CF method, and conducts the recommendation task with the embeddings of users' history behaviors and the corresponding concept, *NAIS* [7] that is also an CF method with an attention mechanism to distinguishe the weights of different online learning behaviors, *ACKRec* [16] that is an attentional graph neural network in a heterogeneous view. For MLP, FISM, and NAIS, we construct the rating matrix and interaction histories between users and concepts from datasets. For ACKRec, we construct the corresponding features and adjacency matrices as inputs based on its steps. We select the most appropriate parameters to obtain the best results for a fair comparison.

From Tabel IV , it is apparent that the performance of DCCR is much better than MLP, FISM, NAIS in both datasets. The AUC of DCCR is about 5.15% to 8.53% higher than MLP, FISM, and NAIS in MOOCCube, and 7.72% to 10.51% higher in XuetangX. Compared with ACKRec, DCCR extracts the correlation feature of user preference, which leads to a better performance. The AUC, HR@20, $NDCG$@20 and MRR of DCCR are 2.06%, 6.22%, 3.62% and 4.7% higher than ACKRec in MOOCCube, respectively, and 3.51%, 10.39%, 9.73%, 4.92% higher in XuetangX. It has a lager growth from ACKRec to DCCR in XuetangX than in MOOCCube, which means that the correlation information in XuetangX is more abundant so that DCCR can make a bigger improvement.

TABLE IV: Results of different methods in Mooccube

Methods	HR			$NDCG$			MRR	AUC
	@5	@10	@20	@5	@10	@20		
MOOCCube								
MLP [2]	0.4335	0.5744	0.7102	0.3562	0.3807	0.4088	0.3335	0.8651
$FISM$ [1]	0.5285	0.7411	0.7715	0.4826	0.5033	0.5288	0.4701	0.8684
$NAIS$ [7]	0.4957	0.6235	0.8497	0.2848	0.3651	0.4218	0.3563	0.8979
$ACKRec$ [16]	0.7125	0.8014	0.8827	0.6326	0.6622	0.6855	0.6015	0.9288
DCCR	**0.7542**	**0.8539**	**0.9297**	**0.6708**	**0.7026**	**0.7217**	**0.6637**	**0.9494**
XuetangX								
MLP [2]	0.3680	0.5899	0.7237	0.2231	0.2926	0.3441	0.2146	0.8595
$FISM$ [1]	0.5849	0.7489	0.7610	0.3760	0.4203	0.4279	0.3293	0.8532
$NAIS$ [7]	0.4112	0.6624	0.8649	0.2392	0.3201	0.3793	0.2392	0.8863
$ACKRec$ [16]	0.6470	0.8122	0.9255	0.4635	0.5170	0.5459	0.4352	0.9232
DCCR	**0.7851**	**0.9063**	**0.9747**	**0.5833**	**0.6259**	**0.6432**	**0.5491**	**0.9583**

IV. Conclusions

This paper proposes a recommendation method named as DCCR which generates the knowledge concept recommendation list for users in MOOCs. DCCR captures the rich entity interactive information in MOOCs by the guide of meta-path and extracts the semantic features from text information along with concepts. It also gathers the correlation features of users among meta-paths by constructing a multi-relational graph. To better consider the association between different meta-paths, a concatenation-based fusion function is proposed to generate the final joint representation of users and concepts. By verifying the effectiveness of DCCR on two public datasets, the experimental results show that DCCR is superior to the existing methods.

References

[1] S. Kabbur, X. Ning, G. Karypis, Fism: factored item similarity models for top-n recommender systems, In: ACM SIGKDD, 2013, pp. 659–667.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.S. Chua, Neural collaborative filtering, In: International Conference on World Wide Web, 2017, pp. 173–182.

[3] K. Järvelin, J. Kekäläinenm, Ir evaluation methods for retrieving highly relevant documents, In: ACM SIGKDD, 2017, pp. 41-48.

[4] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, In: IEEE International Joint Conference on Neural Networks, 2005, vol. 2, pp. 729–734.

[5] L. Pan, X. Wang, C. Li, J. Li, J. Tang, Course concept extraction in moocs via embedding-based graph propagation, In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2017, pp. 875-884.

[6] A. Elbadrawy, G. Karypis, Domain-aware grade prediction and top-n course recommendation, In: ACM Conference on Recommender Systems, 2016, pp. 183–190.

[7] X. He, Z. He, J. Song, Z. Liu, Y.G. Jiang, T.S. Chua, Nais: Neural attentive item similarity model for recommendation, IEEE Transactions on Knowledge and Data Engineering **30**(12), 2354–2366, 2018.

[8] L. Pan, C. Li, J. Li, J. Tang, Prerequisite Relation Learning for Concepts in MOOCs, In: Annual Meeting of the Association for Computational Linguistics, 2017, pp. 1447–1456.

[9] Y. Pang, Y. Jin, Y. Zhang, T. Zhu, Collaborative filtering recommendation for mooc application, Computer Applications in Engineering Education **25**(1), 120–128, 2017.

[10] H. Zhang, M. Sun, X. Wang, Z. Song, J. Tang, J. Sun, Smart jump: Automated navigation suggestion for videos in moocs, In: International Conference on World Wide Web Companion,2 2017 pp. 331–339.

[11] M.T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, In: Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421.

[12] C. King, A. Robinson, J. Vickers, Targeted mooc captivates students, Nature **505**(7481), 26–26, 2014.

[13] C. Shi, B. Hu, W.X. Zhao, S.Y. Philip, Heterogeneous information network embedding for recommendation, IEEE Transactions on Knowledge and Data Engineering **31**(2), 357–370, 2018.

[14] C. Shi, B. Hu, W.X. Zhao, P.S. Yu, Leveraging meta-path based context for top-n recommendation with a neural co-attention model, In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1531–1540.

[15] S. Vashishth, S. Sanyal, V. Nitin, P. Talukdar, Composition-based multi-relational graph convolutional networks, In: International Conference on Learning Representations, 2020, pp. 1–15.

[16] J. Gong, S. Wang, J. Wang, W. Feng, H. Peng, J. Tang, P.S. Yu, Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view, In: ACM SIGIR, 2020, pp. 79–88.

[17] J. Yu, G. Luo, T. Xiao, Q. Zhong, Y. Wang, J. Luo, C. Wang, L. Hou, J. Li, Z. Liu, J. Tang, J.: Mooccube: A large-scale data repository for nlp applications in moocs, In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 3135–3142.

[18] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, In: ICLR Workshop, 2013.

[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, In: International Conference on Learning Representations, 2016.

Postoperative MPA-AUC$_{0\text{-}12h}$ Prediction for Kidney Transplant Recipients based on Few-shot Learning

Pan Qiao
Donghua University
Shanghai, China
panqiao@dhu.edu.cn

Yu Xinyu
Donghua University
Shanghai, China
974581233@qq.com

Li Xinyu
Donghua University
Shanghai, China
13918802469@163.com

Shaokun*
Ruijin Hospital affiliated to Shanghai Jiao Tong University Medical College
Shanghai, China
*Correspondence: shaokuntrue@hotmail.com

***Abstract*—Mycophenolic acid (MPA) is a commonly used immunosuppressive drug. The anti-immune rejection effect of mycophenolic acid is closely related to its exposure level in the body. In clinical practice, mycophenolate acid drug exposure level is usually reflected by monitoring the area under the drug-time curve MPA-AUC$_{0\text{-}12h}$. Calculating the MPA-AUC$_{0\text{-}12h}$ requires numerous blood sampling time points. Not only does the medical staff have more work, but patients suffer more as well. Limited sampling strategies (LSS) are generally used to reduce the number of time points. Nevertheless, this method involves complicated calculations and the predictive accuracy is very low for small sample data. A new method of predicting the MPA-AUC$_{0\text{-}12h}$ value is proposed based on the selection of SHAP features with an improved neural network for small sample data. The experimental results show that the average prediction errors of the MPA-AUC$_{0\text{-}12h}$ values on different data sets by our method are better than that of the baseline models.**

Keywords-kidney transplantation; MPA-AUC$_{0\text{-}12h}$; SHAP; affinitynet

I. INTRODUCTION

Mycophenolic acid (MPA) is a kind of immunosuppressant commonly used in the clinic. It is widely used in the prevention and treatment of acute rejection of transplanted organs[1]. Its strong immunosuppressive effect can significantly reduce the incidence of rejection after transplantation[2]. Clinically, the Area Under Curve (AUC) of postoperative mycophenolic acid administration MPA-AUC$_{0\text{-}12h}$ in renal transplantation patients is often monitored to evaluate the postoperative mycophenolic acid exposure in renal transplantation patients[3]. Too low drug exposure level (low MPA-AUC$_{0\text{-}12h}$) will lead to an increase in acute rejection, and too high drug exposure level (high MPA-AUC$_{0\text{-}12h}$) will lead to an increase in the incidence of gastrointestinal reactions and other adverse reactions.

However, the clinical detection of the MPA-AUC$_{0\text{-}12h}$ value is complicated. Medical staff often need to draw peripheral venous blood from patients at 10 time points. Then the plasma MPA concentration was determined by high-pressure liquid chromatography. According to the linear trapezoidal method, doctors calculated the area under the curve during 0-12 hours (MPA-AUC$_{0\text{-}12h}$) after medication[4]. Due to a number of blood sampling time points, the workload of medical staff is heavy, the cost of testing is high, and the patients are also very painful.

To solve the problem of too many clinical blood sampling time points, limited sampling strategies (LSS)[5] are usually used to calculate the value of MPA-AUC$_{0\text{-}12h}$. The method of limited sampling is divided into two steps. The first step is to determine the time point of blood collection. The second step is to estimate the value of MPA-AUC$_{0\text{-}12h}$ by Multiple Linear Regression (MLR). Although this method solves the problem of too many times of blood collection in clinical practice, the calculation process of choosing blood sampling time points is complicated. In the second step, the main method of existing research is to predict the value of MPA-AUC$_{0\text{-}12h}$ with multiple linear regression and an artificial neural network. The problem with these methods is that when the number of blood sampling time points is very small, the dimension of data in the calculation process is low. Moreover, due to the difficulty in obtaining clinical medical sample data, the data quantity is small. Therefore, the accuracy of the finite sampling method based on linear regression and a simple artificial neural network is not high. There are many research methods for this kind of small sample data. However, most of these few-shot learning methods need to be based on similar large data sets. And the model cannot guarantee the stability of prediction under the condition that there are deviations in the sample distribution of the training set and test set. The prediction effect on the test set is poor.

Given the problems existing in the current methods, this paper proposed a method for selecting blood sampling time points based on the SHAP method and a method for predicting the value of MPA-AUC$_{0\text{-}12h}$ after kidney transplantation based on a few-shot learning model. Then improve the AffinityNet model to predict the MPA-AUC$_{0\text{-}12h}$ value by combining the method of causal reweighting. The contributions of this paper are as follows.

DOI reference number:10.18293/SEKE2022-091

(1) We propose a SHAP-based method to choose blood sampling time points. The method reflects the contribution degree of feature to model output by calculating the marginal contribution of each feature, so as to select the most important feature combination for model training. In this paper, the SHAP-based method is used to select 3 or 4 blood sampling time points from 10 points for calculating the MPA-AUC$_{0\text{-}12h}$.

(2) We propose an improved AffinityNet to predict the MPA-AUC$_{0\text{-}12h}$ value for small sample data. Causal reweighting is used to deal with the deviation in the distribution of small sample data on training and test sets. The sample weight obtained by the causality weight algorithm is integrated into the AffinityNet model, which improves the stability of model prediction on the test set. Due to the low data dimension, the gaussian function is used as an attention kernel to reduce the number of aggregation nodes in the attention layer. By improving the feature extraction effect of the attention pooling layer, the KNN model becomes more suitable for low-dimensional vector aggregation.

II. Related Work

A. Selection Method of Blood Sampling Point

Ratain and Vogelzang[6] first proposed the use of a limited sampling strategy combined with linear regression analysis to predict the MPA-AUC$_{0\text{-}12h}$ value. Yichen Jia[7] et al. used the limited sampling strategy to establish a model to predict the MPA value on the data of 36 kidney transplant patients. They finally got the best blood sampling time points for the model at 0h, 3h, 4h, and 8h. Shao Kun et al.[8] used the limited sampling method and multiple linear regression model to predict the MPA-AUC$_{0\text{-}12h}$ based on the data of 108 patients in the early stage. Although this method can reduce the number of blood sampling time points, the calculation process is complicated, and the accuracy of the linear regression method is not high after the data dimension is reduced.

B. Few-sample Learning Method

Wang YX et al.[18] generated virtual data through the data generation method and trained the classification model with the meta-learning method. Santoro A et al.[9]proposed MANN neural network based on the meta-learning method in 2016. In 2018, Howard J et al.[10] proposed ULMFit fine-tuning language model, which fine-tunes the model by changing the learning rate. However, both meta-learning methods and model fine-tuning methods need a large base data set.

Currently, Tianle Ma[11] proposed AffinityNet small sample neural network for medical disease type prediction. However, AffinityNet is suitable for data with higher feature dimensions. In addition, due to the deviation of the distribution of the training set and test set in the small sample data, the prediction of the model on the test set is not stable.

III. Method

We propose a prediction method of MPA-AUC$_{0\text{-}12h}$ value based on SHAP feature selection. The AffinityNet is improved with causal weights to make it more effective for predicting small sample data. The sample weight obtained by the causal weight algorithm on global data is combined with the loss function of the AffinityNet model. The overall flow chart of the proposed method is shown in Figure 1.

Figure1. the Overall framework of the model.

A. Blood Collection Time-point Selection Module

In this paper, a SHAP-based method is proposed to select the features of blood sampling time points data of kidney transplantation patients. Features with larger SHAP values have a greater influence on results [12].

We used the method to select features of blood collection point data. The input data were blood MPA concentration of kidney transplantation patients before $(0h)$ and after $0.5h$, $1h$, $1.5h$, $2h$, $4h$, $6h$, $8h$, $10h$ and $12h$ in one medication cycle. The output is the feature ranking of MPA-AUC$_{0\text{-}12h}$ values predicted by features at different blood sampling time points. SHAP method measures the importance of blood sampling point features by calculating the marginal contribution of each feature to the model prediction. For all feature sets of blood sampling time points $F = \{X_0, X_{0.5}, X_1, X_{1.5}, X_2, X_4, X_6, X_8, X_{10}, X_{12}\}$ and one feature X_i of blood sampling time points, the SHAP-based method trains two models $f(F)$ and $f(F/X_i)$ with feature X_i and without feature X_i respectively. The predicted values $E[f(F)]$ and $E[f(F/X_i)]$ were obtained by the two models respectively. Then, the marginal contribution W_i of the blood sampling point features on all features was

$$W_i = E[f(F)] - E[f(F/\{X_i\})] \quad (1)$$

In actual prediction, we calculated the marginal contribution of feature X_i on all subsets of feature set excluding feature X_i and calculated the average value to obtain the final SHAP value of the feature of a blood collection point. The calculation method is shown in Formula 2

$$w_i = \sum_{S \epsilon F\{x_i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}(f(S \cup \{x_i\}) - f(S)) \quad (2)$$

Where S is the full subset of the sum of the feature sets excluding X_i features. $f(S \cup X_i) - f(S)$ is the marginal contribution of characteristic X_i to a subset. Then the marginal contribution of the X_i feature on all subsets was averaged to obtain the SHAP value of the feature X_i at blood sampling time points. Finally, SHAP values of all blood sampling time points were ranked. Therefore, the features of the highest blood sampling time points were selected for the construction of the Prediction model of the MPA-AUC$_{0\text{-}12h}$ value.

B. MPA-AUC$_{0\text{-}12h}$ Prediction Module

The model prediction consists of the AffinityNet and causal sample weight modules.

1) AffinityNet Model Structure

AffinityNet model is composed of a Feature Attention Layer and several stacked KNN Attention Pooling layers. The feature attention layer assigns corresponding weight values to the features of each blood collection point. Before these blood sampling time points enter the attention feature extraction module of the graph, attention calculation is carried out for blood sampling time points through the attention mechanism.

Let H_i be the vector composed of p blood sampling time points data of patient I after taking medicine. W_q is the attention weight of the qth blood collection point. The sum of weights of p features is 1, which satisfies the following formula

$$\sum_{q=1}^{p} w_q = 1, w_q \geq 0, W = (w_1, w_2, \dots, w_q) \quad (3)$$

Unlike the usual transforms, the feature attention layer performs element-by-element multiplication. The data of P blood sampling time points and corresponding feature weights of patient i after taking medication were used for element counterpoint multiplication. The data $f(h_i)$ of patient i can be represented by Formula 4. Where x is the parallel multiplication of elements. W still satisfies the constraint in formula 3.

$$f(h_i) = W \times h_i \quad (4)$$

The distance d'_{ij} between patients after transformation can be expressed as Formula 5. It can better measure the similarity between patients and achieve a better aggregation effect.

$$d'_{ij} = ||f(h_i) - f(h_j)|| = ||w \times h_i - w \times h_j||^2 \quad (5)$$

The KNN attention pooling layer can stack a large number of graph attention networks (GAT) together. The kidney transplant patient data vector is represented in the form of nodes in AffinityNet. The neighbor nodes of this node are calculated by the vector similarity. The high-dimensional representation of the data vector is represented by the neighbor nodes of the node at this level through the attention mechanism.

Figure 2 describes how the KNN attention pooling layer extracts the data features of patient i. The data vector of mycophenolic acid after the patient i takes the medicine is h_i. In the generated graph network architecture, other patient data vectors similar to patient i are $h_{i-1}, h_{i+1}, h_{i+2}$. These vectors are called h'_i neighbors in the graph structure. Then the high-dimensional data feature h'_i of the *i-th* patient can be represented by $h_{i-1}, h_{i+1}, h_{i+2}$.

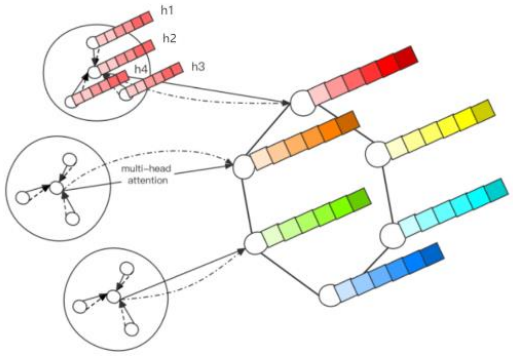


Figure2. The component unit of the attention pooling layer in KNN.

In the graph structure, a node and its nearest m neighbors should have similar feature representations. We employ the GAT module and attention-based pooling to represent vector high-dimensional features. The expression formula is as follows:

$$h'_i = f(\sum_{j \in N(i)} a(h_i, h_j).h_j) \quad (6)$$

Among them, h_i represents the data vector representation of the kidney transplant patient i . h'_i represents the data transformation feature representation of the kidney transplant the i-th patient, and $N(i)$ represents the neighbor of the kidney transplant patient i in the figure. In the KNN attention pooling layer, k is a hyperparameter used to determine how many neighbors of a node will calculate its high-dimensional vector. $f(.)$ is the ReLU() nonlinear activation function combining weight W and bias b:

$$f(h) = \max(Wh + b) \quad (7)$$

$a_{ij} = a(h_i, h_j)$ is the attention calculation after normalization of the i-th and j-th patients. The goal is to calculate the similarity between patients. For introducing the graph attention layer, we use the Gaussian kernel function, which is more suited for low-dimensional vector aggregation. The calculation formula of attention kernel after quoting the Gaussian function is as follows:

$$a_{ij} = a(h_i, h_j) = \frac{e^{-\frac{||h_i - h_j||^2}{2\sigma}}}{\sum_{j \in N(i)} e^{-\frac{||h_i - h_j||^2}{2\sigma}}} \quad (8)$$

2) Causal Sample Weight Model

In causal studies, collinearity between features is an important cause of prediction instability[13]. So that the model cannot learn true causality between features and predicted outcomes. It has been proved in causal studies that under ideal conditions there exists a set of sample weight values that make the original eigenmatrix nearly orthogonal and minimize the collinearity between input variables. Zheyan Shen et al.[14] in 2020 proposed a sample reweighted de-correlation operator to reduce collinearity of input variables to improve collinearity between input matrix features. A Decorrelated Weighting Regression (DWR) algorithm was proposed by Kun Kuang et al. [15] in 2020.

We use the Sample Reweighted Decorrelation Operator (SRDO) algorithm to reduce collinearity between different features of the input matrix. First of all, a design matrix X is used to create an unrelated transformation matrix $\tilde{X}$ according to the column random resampling method. The resulting matrix $\tilde{X}$ breaks the co-distribution among variables in the original matrix X. Then the sample weight is learned by the density ratio estimation method. Study a set of sample weights that make the variable distribution D of X matrix close to distribution $\tilde{X}$ of $\tilde{D}$ matrix.

Specifically, samples in $\tilde{X}$ matrix are labeled as positive samples (Z=1) and samples in the X matrix are labeled as negative samples (Z=0). Fit a probability classifier. According to The Bayesian theory, the density ratio, namely the sample weight, is:

$$w(x) = \frac{p_{\tilde{D}}(x)}{P_D(x)} = \frac{p(x|\tilde{D})}{p(x|D)} = \frac{p(\tilde{D})}{p(D)} \frac{p(Z=1|x)}{p(Z=0|x)} \tag{9}$$

Where x is constant in all samples, so it can be ignored. To find the unit mean of $w(x)$, we can further divide $w(x)'$, the mean of $w(x)$.

$$w(x)' = \frac{1}{n}\sum_{i=1}^{n} w(x_i) \tag{10}$$

$$w(x) = \frac{w(x)}{w(x)'} \tag{11}$$

After the sample weight is obtained, it is counterbalanced by the loss value of the model to correct the loss value of the model. MSE is used as the Loss function in our regression task. The final combination of weights is as follows:

$$loss = \sum_{i=1}^{n}((y_i - y_i^{pred})^2 \times w_i) \tag{12}$$

The specific causal sample weight and model integration process are shown in Figure 3:

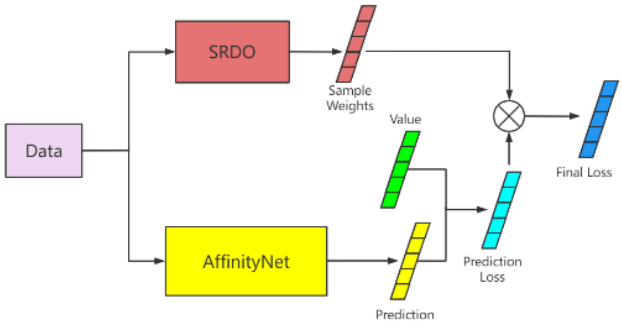


Figure3. The combination of causal sample weight and AffinityNet

IV. EXPERIMENT

A. Dataset

The dataset contains 152 kidney transplant patients. The data of each patient included the features of 10 blood sampling points and the MPA-AUC$_{0\text{-}12h}$ values. The 10 blood sampling time points were the blood mycophenolic acid concentrations collected before medication ($0h$) and $0.5h$, $1h$, $1.5h$, $2h$, $4h$, $6h$, $8h$, $10h$ and $12h$ after medication within one medication interval. The patient's MPA-AUC$_{0\text{-}12h}$ value was the outcome index to be predicted.

B. Blood Collection Time-point Selection

1) Select blood sampling point by SHAP:

SHAP values and importance rankings of different blood sampling time points are shown in Table 1. According to the results in the table, the importance of blood sampling time points from high to low is $2h$, $6h$, $4h$, $1.5h$, $8h$, $1h$, $10h$, $0.5h$, $12h$, $0h$.

TABLE I RANKING OF THE FEATURE IMPORTANCE

Rank	*Feature*	*SHAP Value*
1	T_2	6.55
2	T_6	5.07
3	T_4	4.53
4	$T_{1.5}$	3.30
5	T_8	3.28
6	T_1	3.18
7	T_10	3.15
8	$T_{0.5}$	2.91
9	T_{12}	1.51
10	T_0	0.62

The features of different blood sampling times are selected. The MPA-AUC$_{0\text{-}12h}$ values were predicted by the original AffinityNet. In each group, 20% of the data were randomly selected as the training set. The MPE(Mean Predict Error) values are shown in Table 2.

TABLE II FEATURE SELECTION COMPARISON

	$MPA-AUC_{0-12h}$**(portion = 0.2)**	
Data	train set/31	test set/121
T2,T6	17.042	25.390
T2,T4,T6	14.021	21.587
T1.5,T2,T4,T6	13.014	20.794
T1.5,T2,T4,T6,T8	12.176	18.253
T1,T1.5,T2,T4,T6,T8	12.098	18.638

The MPE of the few-shot learning model decreases with the continuous addition of new blood collection point features. The final blood sampling time points T2, T4 and T6 were chosen for the purposes of ensuring accuracy and reducing the number of features as much as possible.

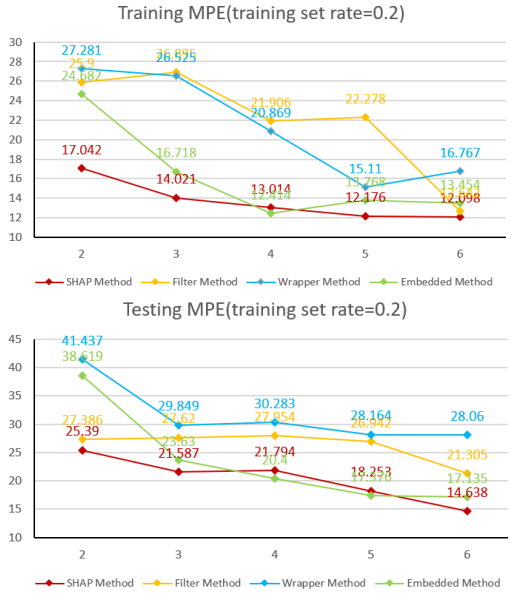


Figure4. MPE comparison of different feature selection methods

2) Comparison of different feature selection methods:

The SHAP-based method was compared with traditional feature selection methods including the Pearson, the Recursive Feature Elimination (RFE), and the Random Forest. As shown in Figure 4, features extracted by SHAP have better effects through model training. The prediction effect is better than other feature selection methods in most cases. When the features increase, the model MPE shows a steady decline.

C. MPA-AUC$_{0\text{-}12h}$ Prediction

1) Model parameter tuning:

a. Aggregate function experiment

Given the reduced dimension of kidney transplantation data after feature selection, this model uses a vector similarity calculation method that is more suitable for low-latitude data as the core of attention. Observe the influence of different aggregation functions on the prediction effect of the original AffinityNet model. The features used in the experiment are the features of $T2$, $T4$ and $T6$ blood sampling time points selected in the previous experiment. The specific comparison results are shown in Table 3.

TABLE III MPE WITH DIFFERENT AGGREGATION FUNCTIONS

	MPE (portion = 0.2)		MPE (portion = 0.1)	
Similarity calculation function	train set	test set	train set	test set
affine	**14.021**	21.587	15.192	38.456
gaussian	14.087	**18.693**	**14.265**	**24.272**
inner-product	18.076	24.645	19.032	31.662
avg$_pool$	19.653	29.435	21.386	36.498
cosine	19.545	30.094	26.767	43.018
key-value	20.417	32.984	25.477	37.950

It can be seen from Table 3 that when 10% of the data is used as training data, the model that uses Gaussian as the attention core realizes the lowest MPE on both the training set and the test set. When 20% of the data is used as training data, using Gaussian as the attention core model has the lowest MPE on the test set, but it is slightly worse than Affine as the attention core on the training set. Therefore, in general, the Gaussian kernel function is the best attention aggregation function of the AffinityNet model in this dataset.

b. Aggregate node number experiment

In order to improve the attention pooling layer's ability to aggregate similar functions, this paper proposes to reduce the KNN attention layer to aggregate neighbor nodes. An experiment is designed to compare the model prediction results of different aggregation nodes, as shown in Figure 5.

When the K value is 3, the model effect is the best. As the K value increases, the model effect gradually becomes worse. This is because when the amount of data samples is small, the KNN attention pooling layer is extracting the value of a certain node. In the feature, the high-dimensional features of the node can be better represented by fewer neighbor nodes, and the learning efficiency can be improved.

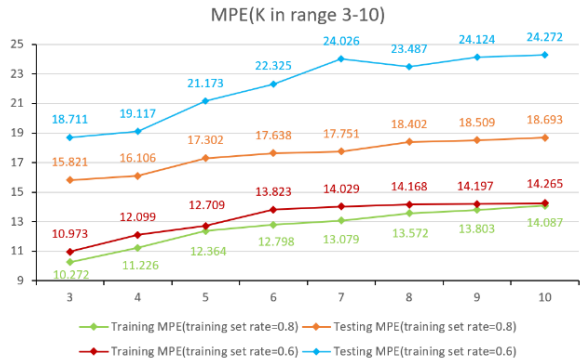


Figure5. The influence of K value selection on the model

2) Comparison of model prediction results:

We use traditional machine learning methods, basic neural networks, and the original AffinityNet model as the baseline model. The comparison results of different models are shown in Table 4.

TABLE IV COMPARISON OF PREDICTION MODELS

	MPE (20% data for training)		MPE (10% data for training)	
predictive model	train set	test set	train set	test set
Logic Regression	26.84	45.54	29.88	49.62
Random Forest	25.24	49.06	28.37	52.33
SVM	23.99	43.09	30.01	51.84
ANN	17.81	32.96	21.57	36.48
CNN	48.59	79.16	60.60	79.12
AffinityNet	10.27	15.82	10.97	18.71
Ours	**8.59**	**10.65**	**8.98**	**14.01**

As can be seen from Table 4, when the number of samples is small, the prediction effect of traditional machine learning methods on test sets is poor. The effect is not good when the sample size is small and the distribution of the training set and test set is different. The prediction effect of the simple ANN and CNN model is not as good as the AffinityNet model. The features of the data can not be better extracted and learned. Our improved AffinityNet has the best prediction effect among all models. Especially in the test set, the prediction effect was significantly improved. Compared with the AffinityNet model, the model with causal weight has no obvious improvement in the training set. When 20% of the samples were used as training samples, the MPE value decreased from 10.27% to 8.59%. When 10% of the samples were used as training samples, the MPE value decreased from 10.97% to 8.98%. However, the effect on the test set is significantly improved. When 20% samples were used as training samples, the MPE of the original model test set was reduced from 15.82% to 10.65%. When 10% samples were used as training samples, the MPE of the original model was reduced from 18.71% to 14.01%. This shows that the model achieves a better learning effect after adding causal weight. The stability of prediction is greatly improved when the distribution of the training set and test set is different.

D. External Data Verification

In order to prove the generalization of our proposed method of predicting MPA-AUC$_{0\text{-}12h}$ after kidney transplantation based on SHAP feature selection and improved AffinityNet model, this section uses the kidney transplant patient data set provided by

another tertiary hospital as an external the data undergoes model validation.

Data from the second hospital dataset included blood mycophenolic acid concentrations at 10 time points and patients' MPA-AUC$_{0\text{-}12h}$ values of 40 kidney transplant patients. The 10 blood sampling time points included before medication ($0h$) and after medication $0.5h$, $1h$, $1.5h$, $2h$, $4h$, $6h$, $8h$, $10h$ and $12h$. MPA-AUC$_{0\text{-}12h}$ is the target value to be predicted.

We used the SHAP-based method for feature selection of blood sampling time points from the data of kidney transplant patients in the second hospital. Then the improved AffinityNet model proposed in this paper is used for prediction. The selection results are shown in Table 5:

TABLE V RANKING OF BLOOD SAMPLING TIME POINTS

Rank	Feature Name	SHAP Value
1	T_4	5.795
2	T_6	5.678
3	T_8	4.914
4	T_3	3.052
5	$T_{2.5}$	1.278
6	T_2	1.150
7	T_12	1.082
8	$T_{1.5}$	1.008
9	T_1	0.488
10	$T_0.5$	0.426
11	T_0	0.376

It can be seen from the results in the table that the feature importance ranking of 10 blood sampling time points for MPA-AUC$_{0\text{-}12h}$ value prediction is $4h$, $6h$, $8h$, $3h$, $2.5h$, $2h$, $12h$, $1.5h$, $1h$, $0.5h$, $0h$ from high to low. We also selected the features of the three most important blood sampling time points at 4h, 6h, and 8h to construct the prediction model. Since the data set of the second hospital had less data, we used 80% of the data as a training set and 20% as a test set. The prediction effect of the improved AffinityNet model proposed in this paper is compared with that of other baseline models. The comparison results are shown in Table 6:

TABLE VI MODEL PREDICTION EFFECT COMPARISON

Prediction Model	Train Set MPE	Test Set MPE
Logic Regression	29.73	48.55
Random Forest	24.25	46.96
SVM	28.90	49.04
ANN	19.89	36.90
CNN	44.55	59.06
AffinityNet	12.68	18.36
Ours	**10.54**	**14.22**

As can be seen from the results in Table 6, the improved AffinityNet neural network proposed in this paper can also achieve good prediction results in other data sets. And the prediction effect of the model is better than all baseline models. The training set MPE reached 10.54%. The MPE can reach 14.22%. Compared with the original AffinityNet, the MPE on the test set is reduced by 22.55%.

V. CONCLUSION

In this paper, a blood collection point selection method based on SHAP is proposed, and the AffinityNet model is improved with causal weight to complete the prediction of MPA-AUC$_{0\text{-}12h}$ The experimental results show that compared with the previous method, the method presented in this paper can achieve better prediction effect, and can effectively reduce the number of clinical blood sampling. It reduces the workload of clinicians and makes the predicted MPA-AUC$_{0\text{-}12h}$ values have greater clinical reference value.

ACKNOWLEDGEMENTS

This work was supported by the National Key RD Program of China under Grant 2019YFE0190500.

REFERENCES

[1] Scott LJ, McKeage K, Keam SJ, et al. Tacrolimus: a further update of its use in the management of organ transplantation[J]. Drugs, 2003, 63(12): 1247-1297

[2] Lu Y P, Zhu Y C, Liang M Z, et al. Therapeutic drug monitoring of mycophenolic acid can be used as predictor of clinical events for kidney transplant recipients treated with mycophenolate mofetil[J]. Transplantation Proceedings, 2006, 38(7): 2048-2050.

[3] Le Meur Y, Buchler M, Thierry A, et al. Individualized mycophenolate mofetil dosing based on drug exposure significantly improves patient outcomes after renal transplantation[J]. Am J Transplant, 2007, 7(1): 2496-2503.

[4] Ren B, Li MW, Tang L, et al. Rapid determination of mycophenolic acid in plasma by HPLC[J]. Chin Hosp Pharm, 2008, 28: 407−408.

[5] Willis C, Taylor PJ, Salm P, et al. Evaluation of limited sampling strategies for estimation of 12-hour mycophenolic acid area under the plasma concentration-time curve in adult renal transplant patients[J]. Ther Drug Monit, 2000, 22: 549-554.

[6] Ratain,M.J. and Vogelzang,N.J.(1987). Limited sampling model for vinblastine pharmacokinetics. Cancer Treat. Rep. 71, 935–939.

[7] Jia Y, Peng B, Li L, et al. Estimation of mycophenolic acid area under the curve with limited-sampling strategy in Chinese renal transplant recipients receiving enteric-coated mycophenolate sodium[J]. Ther Drug Monit, 2017, 39(1): 29-36.

[8] Shao Kun,et al. The relationship between the pharmacokinetics of mycophenolate mofetil and the polymorphism of multidrug resistance gene 1[J]. Chinese Journal of Organ Transplantation, 2009(02) : 81-84.

[9] Santoro A, Bartunov S, Botvinick M, et al. One-shot learning with memory-augmented neural networks. arXiv preprint arXiv: 1605.06065,2016.

[10] Howard J, Ruder S. Universal language model fine-tuning for text classification. arXiv:1801.06146,2018.

[11] Ma T, Zhang A. AffinityNet: semi-supervised few-shot learning for disease type prediction[J]. 2018, 33(01): 1069-1076.

[12] Futagami Katsuya, Fukazawa Yusuke, Kapoor Nakul, KIto Tomomi. Pairwise acquisition prediction with SHAP value interpretation[J]. The Journal of Finance and Data Science. 10.1016/J.JFDS.2021.02.001.

[13] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. arXiv preprintarXiv: 1907.02893,2019.

[14] Zheyan Shen, Peng Cui, Tong Zhang, and Kun Kuang. Stable learning via sample reweighting. AAAI. 5692–5699,2020.

[15] K. Kuang, R. Xiong, P. Cui, S. Athey, and B. Li. Stable prediction with model misspecification and agnostic distribution shift. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 4, 2020:4485–4492

Correlation Feature Mining Model Based on Dual Attention for Feature Envy Detection

1st Shuxin Zhao
Beijing Institute of Technology
School of Computer Science
Beijing, China
zhaosx@bit.edu.cn

2nd Chongyang Shi*
Beijing Institute of Technology
School of Computer Science
Beijing, China
cy_shi@bit.edu.cn

3rd Shaojun Ren
Beijing Institute of Technology
School of Computer Science
Beijing, China
3120191036@bit.edu.cn

4th Hufsa Mohsin
Beijing Institute of Technology
School of Computer Science
Beijing, China
hufsa.bit@yahoo.com

***Abstract*—Feature Envy is a code smell due to the abnormal calling relationships between methods and classes, which adversely affects software scalability and maintainability. Existing methods mainly use various technologies to model abnormal relationships to detect feature envy. However, these methods only rely on local features such as entity names, which is not robust enough. Moreover, the mining depth of correlation features between entities involved in feature envy is limited. In this paper, we propose a correlation feature mining model based on dual attention to detect feature envy. Firstly, we propose a multi-view-based entity representation strategy, which enhanced the robustness of the model while improving the suitability of the correlation feature and model. Secondly, we add attention mechanism to the channel dimension and spatial dimension of CNN to control the flow of information and capture the correlation features between entities more accurately. Finally, the evaluation results on projects both with and without feature envy injected show that our proposed approach outperforms the state-of-the-art methods.**

***Index Terms*—Code Smell, Feature Envy, Software Refactoring, Attention Mechanism, Deep Learning**

I. INTRODUCTION

A code smell is a potential problem in code caused by non-standard programming [1], [2]. Feature envy is a common code smell that has a significant impact on the degree of coupling and cohesion of software [3]–[5]. An accurate and widely accepted definition, first proposed by Beck and Fowler [6], is *more interested in a class other than the one it actually is in*. Based on this definition, many methods have been proposed to complete the critical step of the refactoring operation, that is, the detection of feature envy [7], [8].

The core of the existing feature envy detection methods is to model the abnormal calling relationship between methods and classes, which can be divided into the traditional method based on structural information (code metrics) [8] and the deep learning method based on text information [9]–[11]. The code metrics represent the element overlap degree between the method and the class, but this method relies heavily on artificial design features and selection threshold. In the case that the coding specification is met, there will be a certain correlation between the name of the method and the name of the class. Therefore, many deep learning methods are proposed to automate the end-to-end complex feature mapping [11].

Although deep learning methods based on text information have achieved good performance in feature envy detection, there are two key problems with such methods: 1) the selected local feature of entity name cannot fully represent the entity [11], [12], and the robustness of the model is also affected by the singleness of the feature, for example, when the method name conforms to the specification but the called variable is completely contained in another class. 2) Existing methods usually use CNN or RNN to extract correlation features [11], [13], but they cannot accurately capture effective information and filter other information, resulting in limited expression the ability of the model.

To solve the above problems, we first propose a multi-view based entity representation strategy, which selected name, context and content to represent the entity [14], [15]. The above entity representation strategy was mainly based on the following three observations:

- **Name** is usually an accurate summary of the entity's function, and the method name in the right place should have some correlation with the class name.
- **Context** refers to the external inputs of a method and its outputs to the external. For a method, the inputs are its parameters and the outputs are its return values. In a class with good cohesion, the methods contained in it must be similar in function or goal, which means that the context of multiple methods is similar.
- **Content** of a method includes the external properties and methods called by the method, and two methods that are similar in function are also similar in content. And for method names, which are intuitive generalizations of method functions, we can find deeper correlations between method functions due to the fine-grained nature of the content.

To sum up, we should select information from the three views of method, contain class and target class to get a comprehensive representation. In addition, in order to more accurately capture the correlation features in the selected text

DOI reference number:10.18293/SEKE2022-009

This work is supported by the National Key Research and Development Program of China(No. 2018YFB1003903), National Natural Science Foundation of China (No. 61502033) and the Fundamental Research Funds for the Central Universities.

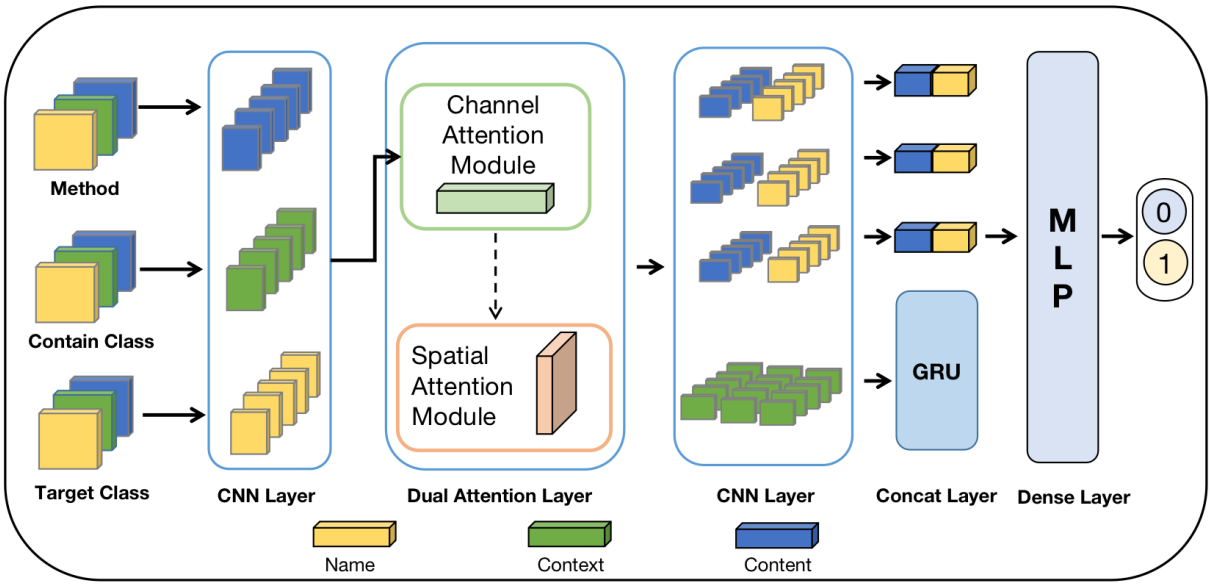


Fig. 1. Data generation process.

information, we propose a correlation feature mining model based on dual attention. The framework is illustrated in Fig.1. Inspired by CBAM [16], this model adds attention mechanism to channel dimension and spatial dimension of feature map obtained by CNN. It can assign more weight to the important feature and the level of the important feature, so the local correlation feature can be accurately captured and filtered. Finally, we use GRU to combine the context among the three entities to obtain the overall correlation from the global view.

The evaluation of the proposed method consists of two parts. 1) On an existing large-scale data-set, which was obtained by manually injecting feature envy through the operation of move methods on seven high-quality open-source Java projects [11]. On this data-set, our method can reach F-measure 55%, which is higher than the state-of-the-art. 2) We tested the proposed method on three open-source Java projects without feature envy injection, and the results showed that the performance of our method is still higher than the existing tools and technologies. The paper makes the following contributions:

- We propose a multi-view based entity representation strategy, which extracts text information from name, context and content to comprehensively represent the entity, so that the entity integrates more correlation features and improves the robustness of the model.
- We propose a dual attention based correlation feature mining model for feature envy detection, based on the comprehensive representation of entities. Dual-channel attention mechanism can expand the depth of text information and accurately filter and capture correlation features.
- The evaluation results on open source projects with and without feature envy injection show that our approach achieves better performance than state-of-the-art approaches.

The remainder of this paper is organized as follows. Section II introduces the work related to feature envy detection. Section III explains the proposed approach, after which Section IV presents the results of the proposed approach. Finally, the conclusions are drawn in Section V.

II. RELATED WORK

Feature envy is a common code smell characterized by being *more interested in a class other than the one it actually is in* [9]. Many methods have been proposed to detect this code smell, including the traditional method based on structural information (code metrics) and the deep learning method based on text information.

Existing feature envy detection methods rely primarily on a metrics that can measure the relationship between the method entity and the class entity [17], [18], which was first proposed by Simon et al. [19] in 2001 . They propose a metric based on the set operation to indicate the distance between entities, as follows:

$$distance(e_1, e_2) = 1 - \frac{|p(e_1) \cap p(e_2)|}{|p(e_1) \cup p(e_2)|} \tag{1}$$

The change rules of $p(e)$ with the entity types of e are as follows:

$$p(e) = \begin{cases} \{e, entities_{Called}\}, & if\ e\ is\ method \\ \{e, entities_{Access}\}, & if\ e\ is\ attribute \end{cases} \tag{2}$$

Entities in code are divided into method entities and attribute entities, where e represents an entity. Here, $P(e)$ represents the set of properties possessed by e. If e is a method, the set contains the entity itself, along with the attribute and method entities that are called by e. If e is an attribute, the set contains the entity itself and all methods that directly access e. Based on the filtered entity set and calculation formula (1)(2),

the distance between entities can be obtained. If the distance between a method entity and the entities in its class is greater than the distance between the entity and the entities in other classes, this method is associated with feature envy.

Tsantalis et al. [20] propose a new metric to define the distance between entities, that differs from that proposed by Simon et al. Although they divide entities into methods and attributes, the final result is the distance between method entities and classes; by contrast, Simon et al. focuses on the distance between method entities and attribute entities [19]. If the method entity m being detected belongs to the class entity C, the formula for calculating the distance is as follows :

$$distance(m, C) = 1 - \frac{S_m \cap S_C}{S_m \cup S_C}, where\ S_C = \bigcup_{e_i \epsilon C} \{e_i\} \quad (3)$$

Otherwise, the distance is computed as follows:

$$distance(m, C) = 1 - \frac{S_m \cap S_C^{'}}{S_m \cup S_C^{'}}, where\ S_C^{'} = S_C \backslash \{m\} \quad (4)$$

In formula (3)(4), m represents a method entity, S_m represents the collection of entities called by the method entity, and S_C represents the method and attribute entities contained in the class; moreover, the measured method entities should be excluded from the collection of their classes. If the final result shows that the distance between a method and the containing class exceeds the distance between the method and the target class, then the method is deemed to be associated with feature envy. This method is implemented by $JDeodorant$, a well-known code smell detection tool, has become the most commonly used benchmark in the code smell detection research field [11].

To make better use of metric information and text information, Liu et al. propose feature envy detection based on deep learning [11], [13]. This method can automatically extract the text information and metrics required by the training classifier from the open-source applications; here the metrics information is the distance metrics proposed by Tsantalis et al. [20], [21], [22]. The text information mainly includes the identifier of the method and the corresponding identifiers of the containing class and the target class [23], [24]. The classifier primarily uses the CNN neural network structure to extract the internal features of the input information. Finally, the extracted features are spliced into the linear layer to predict whether the method and target class is "smelly" or "non-smelly". They are the first to apply deep learning techniques to feature envy detection, and the detection effect is much higher than other methods.

III. METHODOLOGY

A. Data Generation

As we employ deep learning technology to build a mapping between input information containing comprehensive features and feature envy judgment, we need large-scale data to train the model. However, due to the characteristics of code smell, it is difficult to compile relevant training data on a large scale, making it necessary to artificially inject feature envy into the code to generate such large-scale data. We here utilize the method of automatic large-scale data generation proposed by Liu et al. [11].

Finally, we can obtain any number of positive and negative items, as shown in formula (5).

$$Item = < Input, Output > \quad (5)$$

$$Input = < input_m, input_C, input_T > \quad (6)$$

$$Output = < 0/1 > \quad (7)$$

Respectively, the $input_m$, $input_C$, and $input_T$ elements of $input$ represent the information extracted from the movable method m, the containing class C, and the target class T. This information comprises three main parts: name, context, and content, as shown in the following formula (8)(9)(10). The $Output$ information 0 / 1 respectively represents whether this item is negative or positive, as shown in formula (7).

$$input_m = < name(m), context(m), content(m) > \quad (8)$$

$$input_C = < name(C), context(C), content(C) > \quad (9)$$

$$input_T = < name(T), context(T), content(T) > \quad (10)$$

B. Information Processing

Based on the sample generation method, we can get text information that contains correlation features. However, in order to meet the input specifications of the neural network and make better use of data, we still need to process the data.

1) Text Processing: The information we obtain is composed of many identifiers, each of which is generally a combination of one or more words. Thus, to change the input into a form acceptable to the neural network, we need to do the following [25]:

- Divide the identifier into a sequence of words according to the camel case method based on lower-line change, uppercase letters, and numbers.
- Change all words to lowercase.
- Remove programming keywords, special characters and English stop words.

2) Information Combination: Among the three views emphasized by representation strategy, name and content indicate performance of different levels of function, for its part, the context focuses on the input and output of a method, which are the most intuitive representation of a method's interaction with the external environment. Therefore, we combine the name and content modules to represent the internal functional features of a method or class, while the context modules of the method and class are combined to explore the features of the interaction among the method, containing class and target class. By using this combination method, we get a comprehensive representation of entities, and obtain the combination information which is beneficial to correlation mining. We verify the effectiveness of this combination method for feature envy detection in Section IV.

C. Correlation Feature Mining

After the above processing, we have nine unordered sets of words from name, context, and content of three entities. Since the structure is the same, we chose the method name as an example to illustrate our model flow.

First, we obtain a dense embedding matrix $X \in R^{N \times d}$ by embedding the sequence of words forming method name through the embedding space, where N is the number of words and d is the dimension. Most of the text information extracted was short text, so we chose CNN (convolutional neural network) with good local feature extraction ability to process X.

$$F = Conv(Emb(X)) \tag{11}$$

As shown in formula (11), the size of three convolution kernels are 2 $\times d$, 3$\times d$ and 4$\times d$ respectively, F is the set of feature maps obtained after convolution.

In order to accurately capture features, inspired by CBAM, we add attention mechanisms in channel and spatial dimensions respectively, where channel attention focuses on the difference in importance of features and spatial attention focuses on the difference in location of features.

$$F' = F \times \sigma(W_1(W_0(F_{avg})) + W_1(W_0(F_{max})) \tag{12}$$

$$F'' = F' \times \sigma(f(F'_{avg}; F'_{max})) \tag{13}$$

The operation is shown in formula (12)(13), and the feature map set F'' containing weights is obtained. We then perform another convolution operation on F'' to capture the deep features.

As mentioned in the above chapter, context interaction is the most direct expression of the correlation of the three entities. Therefore, context information S_m, S_c and S_t of the three entities are input into GRU as state flow. The update process of GRU is as formula (14). The hidden state h_i in formula X contains the correlation features of three contexts.

$$z_i = \sigma(W_z x_i + U_z h_{i-1}) \tag{14}$$

$$r_i = \sigma(W_r x_i + U_r h_{i-1}) \tag{15}$$

$$\tilde{h_i} = tanh(W x_i + U(r_i \odot h_{i-1})) \tag{16}$$

$$h_i = (1 - z_i) h_{i-1} + z_i \tilde{h_i} \tag{17}$$

Finally, to extract the global correlation features, we concatenate the name and content of each entity, that is, F'' obtained in formula (13), the concatenate results and h_i from formula (14) are entered into the fully connected neural network, whose output represents the final classification result: smelly or no-smelly. In addition, we select $binary\_crossentropy$ as loss function, which is defined as follows:

$$L = \sum_{i=1}^{N} y^{(i)} \log \hat{y^{(i)}} + (1 - y^{(i)}) \log(1 - \hat{y^{(i)}}) \tag{18}$$

where $\hat{y^{(i)}}$ is the true type of the method, and $y_{(i)}$ is the prediction of our proposed model.

IV. EXPERIMENTS

A. Research Questions

We evaluate our proposed approach by answering the following research questions.

- **RQ1**: Does our proposed approach outperform the state-of-the-art approaches in detecting feature envy?
- **RQ2**: How does our proposed method perform on real projects without feature envy injected?
- **RQ3**: Are the proposed representation strategy and dual attention mechanism helpful for FE detection?

Both RQ1 and RQ2 focus on the performance difference between our proposed method and other technologies in feature envy detection, so we choose the deep learning-based method proposed by Liu et al. [11] and the two popular code smell detection tools $JDeodorant$ and $JMove$ as comparison methods. RQ3 is mainly to verify the validity of the model. We verify the combination of name, context and content. At the same time, we also conduct an ablation experiment on the dual-attention mechanism.

B. Dataset and Experimental Design

The data used for experimental evaluation can be divided into two parts; (i) Large-scale data with feature envy automatically injected for training and verification of the classifier. (ii) Small-scale data without feature envy injected that is used to evaluate the effectiveness of the proposed approach on real projects.

To avoid over-fitting and reduce the impact of insufficient data size on the detection results, we choose to use the k-fold cross-validation method. Moreover, we selected three commonly used indicators, F_1, $Recall$, and $Precision$, to evaluate the effect of each method (19) (20) (21).

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{19}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{20}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{21}$$

C. RQ1: Detection on Injected Projects

To effectively verify that our proposed approach performs better than the best comparison approach, we select the highest-performance deep learning-based method and two popular code smell detection tools for comparison. The evaluation results are presented in Table I. From the above data, it can be determined that our method outperforms the best existing technology. Specifically, the method we proposed is achieves significantly better results than the traditional tool $JDeodorant$ and $JMove$ on the three indicators. Moreover, compared to the method proposed by Liu et al., which also uses deep learning technology, our method has higher precision but slightly lower recall, and finally, our method performs better in terms of comprehensive indicators.

The improvement of precision indicates that the prediction results of our method are more reliable, which in turn suggests

TABLE I
EVALUATION RESULT ON FEATURE ENVY DETECTION

Applications	Proposed Approach			Approach of Liu			JDeodorant			JMove		
	precision	recall	F1	precision	recall	F1	precision	recall	F1	precision	recall	F1
JUnit	51.90%	100.00%	68.33%	40.59%	91.11%	56.16%	30.76%	14.82%	20%	22.72%	18.52%	20.41%
PMD	67.44%	78.38%	72.50%	41.27%	68.42%	51.49%	15.79%	5.36%	8%	30%	26.79%	28.3%
JExcelAPI	25.52%	68.52%	37.19%	31.9%	92.85%	47.49%	60%	10.7%	18.18%	27.27%	16.07%	20.22%
Areca	38.42%	75.26%	50.87%	46.05%	72.16%	56.23%	32.14%	9.28%	14.4%	26.76%	39.18%	31.8%
Freeplane	63.16%	75.29%	68.69%	38.09%	68.58%	48.97%	21.62%	8.94%	12.65%	24.83%	13.79%	17.73%
jEdit	34.66%	72.73%	46.94%	42.63%	78.57%	55.28%	22.73%	4.55%	7.58%	17.43%	13.57%	15.26%
Weka	35.19%	66.25%	45.97%	40.05%	86%	54.65%	58.33%	17.5%	26.92%	11.22%	11.75%	11.48%
Average	**45.18%**	**76.63%**	**55.79%**	39.79%	79.27%	52.98%	39.51%	12.22%	18.66%	18.37%	16.3%	17.27%

The data of approach of Liu, JDeodorant and Jmove are cited from Deep Learning Based Feature Envy Detection [11].

that the complementary relationship between the three views of representation strategy that we propose has indeed corrected the erroneous preference in prediction. Since a certain contradiction exists between the two indicators of precision and recall, the inconsistency of the two indicators is also within the acceptable range: specifically, precision and recall for our method is 31.45% (76.63%-45.18%), which is much smaller than the 39.48% (79.27%-39.79%) of the method proposed by Liu et al. This further proves that our extraction of feature envy features is highly comprehensive, allowing us to obtain a more reliable and stable detector.

D. RQ2: Detection on Projects without Injection

As shown in Table I, our proposed method outperforms the best technology on java projects that automatically inject feature envy. However, the automatically injected feature envy characteristics must be based on the assumption that there is no misplaced method in this project, which is difficult to fully guarantee. Moreover, the feature envy created by the moving method may have certain key differences when compared to real feature envy, which leads to deviations in the extracted features and affects the effect of the detector. We accordingly choose to verify our method on real projects without injected feature envy to its effectiveness in real-world scenarios. Four graduate students with rich Java development experience will review the test results and provide their opinions on whether the findings can be accepted as true feature envy, and take more than half of the opinions will be taken as the final result. Finally, we conducted feature envy testing on three real projects according to the process. The results are shown in Table II.

From the table, we can construct our method is still superior to other methods on real projects without feature envy injection. Compared with Liu's method, the detection accuracy of our method is 16.97% (58.20%-41.23%) higher. Moreover, our method also achieves performance improvements of 30.61% (58.20%-27.59%) and 42.85% (58.20%-15.35%) relative to $JDeodorant$ and $JMove$ respectively.

E. RQ3: The Effectiveness of Model

To verify the effectiveness of each of the three perspectives, we set up four sets of experiments. (i) The information contains three modules. (ii) Name module deleted; (iii) Context module deleted; (iv) Content module deleted. The results of the experiments are shown in Fig. 2. From the figure, it can

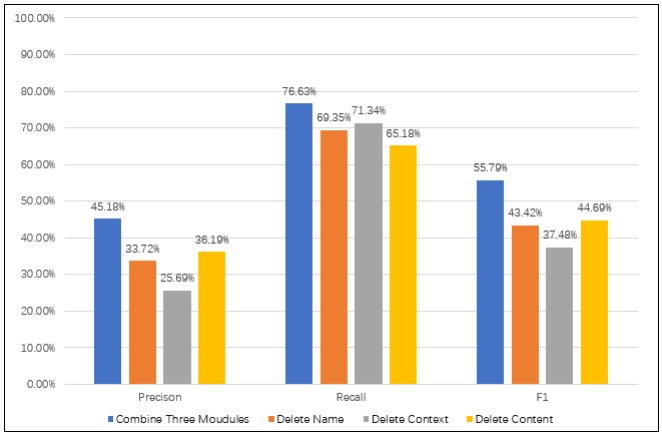


Fig. 2. Results of three perspectives.

be observed that deleting any module will lead to a decline in the detection result, which proves that all three perspectives complement each other and lead to more comprehensive feature representation. After deleting the context module, f1 dropped by 18.31% (55.79%-37.48%), which is the largest drop; this indicates that the context module is most important for feature envy detection.

Inside the neural network, we added a dual attention mechanism on CNN. To prove the effectiveness of this operation, we deleted the operation and got the results as shown in the following Table III:

As can be seen from the table, after deleting the dual attention mechanism, the three indicators of precision, recall, and F_1 were reduced by 4.01% (45.18%-41.17%), 3.67% (76.63%-72.96%), and 3.55% (55.79-52.24%). Therefore, the

TABLE II
EVALUATION RESULT ON PROJECTS WITHOUT INJECTING FEATURE ENVY

Metrics	Proposed Approach				Approach of Liu				JDeodorant				JMove			
	XMD	**JSmooth**	**Neuroph**	**Total**	**XMD**	**JSmooth**	**Neuroph**	**Total**	**XMD**	**JSmooth**	**Neuroph**	**Total**	**XMD**	**JSmooth**	**Neurpoh**	**Total**
Reported	40	22	72	134	32	26	56	114	8	3	18	29	106	27	82	215
Accepted	28	10	40	78	15	11	21	47	3	1	4	8	12	5	16	33
Precision	**70.00%**	**45.45%**	**55.56%**	**58.20%**	46.88%	42.31%	37.5%	41.23%	37.5%	33.33%	22.22%	27.59%	11.32%	18.52%	19.51%	15.35%

The data of approach of Liu, JDeodorant and Jmove are cited from Deep Learning Based Feature Envy Detection [11].

TABLE III
DETECTION RESULT WITHOUT JOINT FEATURES EXTRACTION

Applications	Precision	Recall	F1
JUnit	55.93%	80.49%	66.00%
PMD	52.54%	83.78%	64.58%
JExcelAPI	22.00%	61.11%	32.35%
Areca	41.72%	70.10%	52.31%
Freeplane	51.90%	85.88%	64.70%
jEdit	31.01%	60.61%	41.03%
Weka	33.09%	68.75%	44.68%
Average	**41.17%**	**72.96%**	**52.24%**

dual attention mechanism does enhance the expressive ability of the model.

V. CONCLUSION

In this paper, a correlation feature mining model based on dual attention to detect feature envy is proposed. At first, a new representation strategy is proposed, which extracts the text information from the three views of name, content and context to comprehensively express entities. Secondly, a dual attention mechanism is used to accurately capture local and global correlation features to detect Feature envy. Lastly, We respectively evaluated the method on seven open-source Java projects that automatically injected feature envy and three open-source Java projects that did not inject feature envy. The results show that the feature envy detection effect of the proposed method is indeed superior to the state-of-the-art. Recommending suitable target class for methods with feature envy will be considered as a potential future work.

REFERENCES

[1] A. April and A. Abran, *Software maintenance management: evaluation and continuous improvement*, vol. 67. John Wiley & Sons, 2012.

[2] W. J. Brown, R. C. Malveau, H. W. McCormick III, and T. J. Mowbray, "Refactoring software, architectures, and projects in crisis," 1998.

[3] A. K. Das, S. Yadav, and S. Dhal, "Detecting code smells using deep learning," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pp. 2081–2086, Oct 2019.

[4] T. Sharma and D. Spinellis, "A survey on software smells," *Journal of Systems and Software*, vol. 138, pp. 158–173, 2018.

[5] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *Ieee software*, vol. 29, no. 6, pp. 18–21, 2012.

[6] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Signature Series (Fowler), Pearson Education, 2018.

[7] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. Le Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 20–36, 2009.

[8] R. Marinescu, "Detection strategies: Metrics-based rules for detecting design flaws," in *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, pp. 350–359, IEEE, 2004.

[9] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, "Machine learning techniques for code smell detection: A systematic literature review and meta-analysis," *Information and Software Technology*, vol. 108, pp. 115–138, 2019.

[10] F. Palomba, A. Panichella, A. De Lucia, R. Oliveto, and A. Zaidman, "A textual-based technique for smell detection," in *2016 IEEE 24th international conference on program comprehension (ICPC)*, pp. 1–10, IEEE, 2016.

[11] H. Liu, Z. Xu, and Y. Zou, "Deep learning based feature envy detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 385–396, 2018.

[12] X. Guo, C. Shi, and H. Jiang, "Deep semantic-based feature envy identification," in *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, pp. 1–6, 2019.

[13] H. Liu, J. Jin, Z. Xu, Y. Bu, Y. Zou, and L. Zhang, "Deep learning based code smell detection," *IEEE transactions on Software Engineering*, 2019.

[14] N. Tsantalis and A. Chatzigeorgiou, "Identification of extract method refactoring opportunities for the decomposition of methods," *Journal of Systems and Software*, vol. 84, no. 10, pp. 1757–1782, 2011.

[15] J. Chang and D. M. Blei, "Hierarchical relational models for document networks," *The Annals of Applied Statistics*, pp. 124–150, 2010.

[16] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[17] N. Anquetil and T. C. Lethbridge, "Experiments with clustering as a software remodularization method," in *Sixth Working Conference on Reverse Engineering (Cat. No. PR00303)*, pp. 235–255, IEEE, 1999.

[18] T. Mens, N. Van Eetvelde, S. Demeyer, and D. Janssens, "Formalizing refactorings with graph transformations," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, no. 4, pp. 247–276, 2005.

[19] F. Simon, F. Steinbruckner, and C. Lewerentz, "Metrics based refactoring," in *Proceedings fifth european conference on software maintenance and reengineering*, pp. 30–38, IEEE, 2001.

[20] N. Tsantalis and A. Chatzigeorgiou, "Identification of move method refactoring opportunities," *IEEE Transactions on Software Engineering*, vol. 35, no. 3, pp. 347–367, 2009.

[21] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2014.

[22] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[23] H. Liu, M. Shen, J. Zhu, N. Niu, G. Li, and L. Zhang, "Deep learning based program generation from requirements text: Are we there yet?," *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[24] Y. Jiang, H. Liu, and L. Zhang, "Semantic relation based expansion of abbreviations," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 131–141, 2019.

[25] M. F. Porter, "An algorithm for suffix stripping," *Program*, 1980.

Graph Embedding Models for Community Detection

Yinan Chen*, Zhuanming Gao*, Dong Li+
South China University of Technology, Guangzhou 510006, China

***Abstract*—Graph embedding models, also known as network representation models, have been tried to be applied to community detection tasks. However, most existing graph embedding models are not specially designed for community detection tasks and thus may be incapable of revealing the community structures in networks well. To fill this gap, this paper proposes two novel graph embedding models, GEMod and GEMap, which are specially designed for community detection. The proposed methods try to optimize the modified modularity and two-level coding length while learning the nodes embedding, so that the learned nodes embedding can be better applied to detect community structures in networks. Experimental results show that the algorithms proposed are superior or comparable to other community detection algorithms based on graph embedding models. Besides, the nodes embedding generated by GEMod and GEMap are generally more compact and separable, which means that they are more suitable for clustering tasks.**

Keywords—community detection, graph embedding, clustering

I. INTRODUCTION

Many complex systems exist in the form of networks or can be modeled as networks, such as social networks, scientists collaboration networks, epidemic spreading networks and protein interaction networks. Community detection is an important task in the field of network analysis, which aims to reveal the community structures in networks. A community is generally defined as a group of nodes which are closely connected internally, while the connections between different community nodes are sparse.

The graph embedding task attempts to represent network nodes with low-dimensional continuous vectors and simultaneously capture the structural information of the network. Graph embedding can provide effective input for downstream machine learning tasks, such as node classification [1], link prediction [2] and graph visualization [3]. With the gradual maturity of graph embedding, some scholars try to apply it to community detection tasks [4][5]. However, most existing graph embedding models are not designed for community detection, so they may not be able to effectively detect the community structures in networks.

Inspired by [6] and [7], we modify the definition of modularity and two-level coding length by using the nodes embedding, and propose the GEMod and GEMap graph embedding models. Same as DeepWalk [8] model, GEMod and GEMap are both based on random walk, but they take the community structure into consideration while learning the nodes embedding, so that the learned nodes embedding can be better applied to detect the communities in networks. Specifically, the GEMod model will try to optimize the modified modularity, and the GEMap model will try to optimize the modified coding length. Experimental results show that our methods can generally generate more compact and separable nodes embedding as shown in Fig. 1.

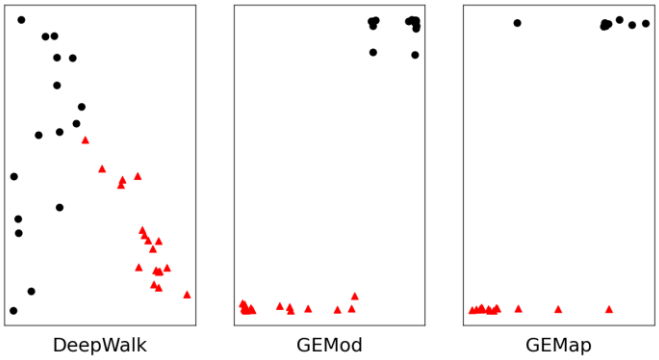


Figure 1. Node embeddings of Karate Club network. Different colors represent different community nodes.

The contributions of this paper are summarized as follows:

- Based on nodes embedding, a modified definition of modularity and two-level coding length are proposed.
- The modified community structure metrics are explicitly introduced into the graph embedding models, so that the learned nodes embedding can be better applied to the community detection tasks.
- The methods proposed can achieve more compact and divisible clustering results.

II. RELATED WORK

A. Community Detection

Newman et al. first introduced the definition of modularity [6] and used it as the evaluation metric of community partition. Specifically, the modularity is defined as follows:

* These authors have contributed equally to this work.
+ Corresponding Author. E-mail: cslidong@scut.edu.cn

DOI reference number: 10.18293/SEKE2022-005

$$Q = \frac{1}{2m}\sum_{i,j}\left[\boldsymbol{A}_{i,j} - \frac{d_i d_j}{2m}\right]\delta(\boldsymbol{C}_i, \boldsymbol{C}_j) \quad (1)$$

where m denotes the number of edges of the network and $\boldsymbol{A}_{i,j}$ denotes the number of edges between node i and node j. d_i and $\boldsymbol{C}_i$ respectively denote the degree of node i and the community that node i is located in. $\delta(\boldsymbol{C}_i, \boldsymbol{C}_j)$ is the Kronecker delta, which equals to 1 if $\boldsymbol{C}_i$ is equal to $\boldsymbol{C}_j$, otherwise 0. Many subsequent community detection algorithms based on modularity optimization have also been proposed, such as [9][10].

Besides the optimization method based on modularity, community detection based on information theory is also a widely studied direction. [7][11] Among them, the Infomap algorithm regards community detection in networks as a problem of map creating, and holds that a good map needs to be well compressed, so that the length of each path in the map should be short as possible. The algorithm uses information entropy to represent the average path length, and proposes the idea of two-level coding to measure the average coding length of random walking in the network. Specifically, the coding length is defined as:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p^i H(P^i) \quad (2)$$

where M represent the partition scheme, $H(Q)$ represents the average coding length between communities, and $H(P^i)$ represents the average coding length of community i, q represents the probability of jumping between different communities, and p^i represents the probability of staying inside community i.

B. Graph Embedding

Bryan et al. proposed the DeepWalk algorithm [8] based on natural language model. The basic idea is to apply the process of random walk for each node in the network to obtain node sequences, then regard each node as a word and node sequences as sentences. After that, based on the SkipGram [12] language model, the low-dimensional vector representation of each node is learned. [13][14][15]

In recent years, network representation algorithms based on graph neural networks have also been proposed, such as [16][17][18]. However, most of them are supervised learning models or semi-supervised learning models, while community detection is an unsupervised learning task. Consequently, these graph neural networks can not be directly applied to community detection in networks.

C. Graph Embedding and Community Detection

An intuitive way of community detection based on network representation is to obtain the nodes embedding of the network by applying some kind of graph embedding model, and then cluster the embeddings by a clustering algorithm [4][5], so as to achieve the goal of community detection. However, in such approach, the network representation process is independent of the node clustering process, and the network representation model cannot get feedback from the nodes clustering model.

In order to alleviate the above problem, the ComE [19] model combines node embedding, community embedding and community detection into a single process, so as to complement each other. However, it assumes that the community embedding obeys a multivariate Gaussian distribution. GEMSEC [20] model introduces a self-clustering process into the nodes embedding process, thus improving the clustering quality of nodes representation, but it does not explicitly introduce community structure metrics.

III. The Methods

A. Problem Definition

The methods mainly focus on detecting non-overlapping communities by using graph embedding methods, given an undirected and unweighted network $G = (V, E)$.

Definition 1 Non-overlapping Community Detection

Given a network $G = (V, E)$, non-overlapping community detection aims to divide V into K disjoint node subsets $\{P_i | P_i \subset V, P_i \cap P_j = \emptyset, i \neq j, i = 1, \ldots, K\}$, and $\cup P_i = V$, so that the nodes in each node subset share some kind of similarity, while different node subsets have great dissimilarity.

Definition 2 Graph Embedding

Given a network $G = (V, E)$, graph embedding models aim to find a mapping function $f: V \to \mathbb{R}^d$, so that the learned nodes embedding can effectively express the structural information of the network. d is the dimension of the embedding space. That is, the nodes are projected from discrete space to a continuous vector space.

B. Node Similarity

Given nodes $u, v \in V$ and mapping function f, let $\boldsymbol{h}_u = f(u)$ and $\boldsymbol{h}_v = f(v)$, $\boldsymbol{h}_u, \boldsymbol{h}_v \in \mathbb{R}^d$. Graph embedding models often use the softmax or sigmoid function to measure the similarity or adjacency probability of u and v. Nevertheless, nodes embedding will generally serve as the input of some kind of clustering model, and many clustering models usually uses Euclidean distance to measure the dissimilarity between different samples. The dissimilarity between node u and node v is defined as:

$$dissim(u, v) = \|\boldsymbol{h}_u - \boldsymbol{h}_v\|_2 \quad (3)$$

The opposite number of the dissimilarity is defined as the similarity measure between nodes:

$$sim(u, v) = -dissim(u, v) \quad (4)$$

C. GEMod Algorithm

The GEMod model includes two stages: embedding initialization and modified modularity optimization. Specifically, the algorithm firstly takes each node as the starting point to do multiple truncated random walks. The process of random walk can be regarded as the process of message propagation. Since the small world effect [21] generally exists in networks, the length of each walk is set to a value less than 6. After that, the nodes in the same walk sequence are regarded as the friend nodes, and let the friend nodes of node u be $F(u)$. It is assumed that the a node

and its friend nodes should have great similarity for they share some kind of characteristic. Similar to the SkipGram model, GEMod also performs negative sampling to obtain another set of node sequences, and takes the nodes in the sequence as stranger nodes of the source node, and let the stranger nodes of node u be $S(u)$. The negative sampling process of GEMod is the same as that of SkipGram model.

GEMod expects to maximize the similarity between node u and its friend nodes, and simultaneously maximize the dissimilarity between node u and its stranger nodes. Consequently, the loss function corresponding to the first stage is:

$$L_1 = -\left[\sum_{u\in V}\sum_{v\in F(u)} sim(u,v) + \sum_{u\in V}\sum_{v'\in S(u)} dissim(u,v')\right] \quad (5)$$

In order to make the node embeddings better reflect the community structures, and make the connections within community closer, while the connections between communities more sparse, a modified definition of modularity is proposed:

$$M = \sum_{u,v\in V} sim(u,v)\delta(\mathbf{C}_u,\mathbf{C}_v) + \sum_{i=1}^{K}\sum_{j=i+1}^{K} dissim(C^{i,0}C^{j,0}) \quad (6)$$

$$h^{i,0} = \frac{1}{|C^i|}\sum_{u\in C^i} h_u \quad (7)$$

where $\mathbf{C}_u$ represents the community to which the node u belongs, C^i represents the i -th community, $C^{i,0}$ represents the center of community i, $h^{i,0}$ is embedding of $C^{i,0}$, K is the number of communities, and $\delta(\mathbf{C}_u,\mathbf{C}_v) = 1$ if $\mathbf{C}_u = \mathbf{C}_v$, otherwise, $\delta(\mathbf{C}_u,\mathbf{C}_v) = 0$. Herein, we use k-means to cluster the nodes embedding to obtain the community partition of the network, and then calculate the modified modularity. It should be pointed out that other clustering methods are also feasible. The meaning of maximizing the above equation is to maximize the similarity of nodes within the same community and the dissimilarity between different community centers. Thus, the connections within communities are tight while the communities are far away from each other. As a result, the nodes embedding generated by GEMod model can get more compact and separable clusters. The loss function of the second stage is,

$$L_2^{mod} = L_1 - \alpha M \quad (8)$$

where α is a hyper-parameter used to balance the influence of M on the result.

In order to accelerate the convergence of the algorithm, GEMod will be trained for a certain number of rounds in the first stage, and then enter the second stage.

D. GEMap Algorithm

The GEMap algorithm is similar to the GEMod algorithm, but the second stage of GEMap tries to optimize the modified coding length instead of the modified modularity. The modified coding length also uses the idea of two-level coding, including coding within communities and coding between communities. However, unlike the Infomap algorithm, GEMap expects to minimize the coding length within communities and maximize the coding length between communities.

Suppose that there is a signal source in the center of each community, which is called a local signal source, and the nodes closer to the signal source have more opportunities to receive the message sent by the signal source. Therefore, the probability that a node receives a message by the distance between the node and the signal source can be measured. Specifically, for the community C^i, the distances between each node in the community and the community center $C^{i,0}$ are first calculated, then divided by the sum of all distances, and finally sorted in descending order to get the probability distribution p^i. $p^{i,1}$ represents the receiving probability of the nearest node from the community center, $p^{i,2}$ represents the receiving probability of the next nearest node from the community center, and so on. Actually, the average coding length of each community has nothing to do with the order of p^i, so the sorting process can be omitted.

Similarly, suppose that there is also a signal source in the center of the network composed of all community centers, which is called the global signal source, and then calculate the probability that each community center receives the message sent by the global signal source as described above, the average coding length between communities can be calculated.

Since we only focus on non-overlapping community detection in networks, we make an assumption that each signal source only produces messages belong to a specific topic, and each community is only interested in a specific topic, while different communities do not share the same interest. Thus, we expect to minimize the average coding length within communities and maximize the average coding length between communities. In summary, the average intra-community coding length of each community is defined as follows:

$$E_{intra} = -\sum_{i=1}^{K}\sum_{u\in C^i} p^{i,u}\log p^{i,u} \quad (9)$$

$$p^{i,u} = \frac{sim(C^{i,u},C^{i,0})}{\sum_{u\in C^i} sim(C^{i,u}C^{i,0})} \quad (10)$$

where $C^{i,u}$ represents the node u in community i. And the average inter-community coding length is defined as follows:

$$E_{inter} = -\sum_{i=1}^{K} q^i \log q^i \quad (11)$$

$$q^i = \frac{sim(C^{i,0},C^0)}{\sum_i^K sim(C^{i,0},C^0)} \quad (12)$$

$$h^0 = \frac{1}{K}\sum_{i}^{K} h^{i,0} \quad (13)$$

where $C^{i,0}$ is the center of community i, C^0 is the centroid of community centers, $h^{i,0}$ is the embedding of $C^{i,0}$, h^0 is the embedding of C^0 and K is the number of communities in the network. And the overall coding length is,

$$E = E_{intra} + E_{inter} \quad (14)$$

In summary, the loss function of the second stage of GEMap algorithm is,

$$L_2^{map} = L_1 + \beta E \tag{15}$$

where β is a hyperparameter used to balance the influence of E on the result.

E. Models Optimization

Both GEMod and GEMap models need to optimize the parameter set of $\mathbf{H} = \{h_u | u \in V\}$, and its size is $O(d|V|)$. Herein, we use the back-propagation algorithm to calculate the derivative of the loss function, and choose the Adam [22] optimizer to optimize the model parameters.

IV. Experimental Results and Analysis

A. Data-sets

In this paper, the effectiveness of the algorithms is verified on four real-world data-sets [23][24][25] and four LFR [26] synthetic data-sets. The specific network structure information of each data-set is shown in Table I. In which n represents the number of nodes, m represents the number of edges, k represents the number of ground-truth communities, d represents the average degree of nodes, and μ represents the mixing parameter for synthetic networks.

TABLE I. Main properties of the data-sets

Data-set	n	m	k	d	μ
Karate	34	78	2	4.6	-
Dolphin	62	162	2	5.1	-
Polbooks	105	441	3	10.7	-
Football	115	613	12	8.4	-
L1	1,000	15,304	49	15	0.3
L2	1,000	30,708	29	30	0.3
L3	1,000	15,206	49	15	0.5
L4	1,000	30,156	31	30	0.5

B. Comparison Algorithms

Here six graph embedding models are selected to compare with GEMod and GEMap algorithms, including DeepWalk [8], Node2vec [13], WALKLETS [14], LINE [15], ComE [19] and GEMSEC [20]. Specifically, the embeddings learned by these models are clustered using k-means, to obtain the community partition for a network.

C. Evaluation Metric

Because the data-sets have ground-truth community partition, normalized mutual information (NMI) [27] is used to measure the similarity between the partition output by algorithm and the ground-truth partition, which is defined as follows:

$$NMI = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} C_{ij}\log_2\left(C_{ij}N/C_{i.}C_{.j}\right)}{\sum_{i=1}^{C_A} C_{i.}\log_2(C_{i.}/N) + \sum_{j=1}^{C_B} C_{.j}\log_2\left(C_{.j}/N\right)} \tag{16}$$

where C_A and C_B respectively represents the community partition obtained by the algorithm and the ground-truth community partition, and C_A and C_B respectively represents the number of communities in partition C_A and partition B. C is the confusion matrix, and C_{ij} represents the number of nodes in the community i divided by C_A and also in the community j divided by B. $C_{i.}$ represents the sum of elements in the i -th row of the confusion matrix, $C_{.j}$ represents the sum of elements in the j -th column of the confusion matrix, and N is the total number of nodes in the network. The value range of NMI is $[0,1]$. The larger the NMI value, the closer the partition result obtained by the algorithm is to the ground-truth community partition.

D. Experimental Results

Each algorithm runs five times on each data-set, and finally take the average of the results. The comparison results of the algorithms are shown in Table II and Table III. The error of the experimental results is indicated in parentheses, which is measured by the standard deviation of the results.

TABLE II. Average NMI of each algorithm on real-world networks

Data-set	Karate	Dolphins	Polbooks	Football
DeepWalk	0.663 (±0.038)	0.817 (±0.048)	0.562 (±0.003)	0.925 (±0.001)
Node2vec	0.946 (±0.120)	0.874 (±0.033)	0.561 (±0.024)	0.927 (±0.002)
WALKLETS	0.869 (±0.073)	**0.889** (±0.000)	0.557 (±0.012)	0.927 (±0.000)
LINE	0.473 (±0.103)	0.322 (±0.131)	0.409 (±0.045)	0.852 (±0.020)
ComE	0.604 (±0.049)	0.453 (±0.019)	0.465 (±0.035)	0.691 (±0.117)
GEMSEC	0.226 (±0.000)	0.293 (±0.000)	0.103 (±0.000)	**0.930** (±0.000)
GEMod	**1.000** (±0.000)	**0.889** (±0.000)	0.568 (±0.007)	0.927 (±0.001)
GEMap	**1.000** (±0.000)	**0.889** (±0.000)	**0.573** (±0.009)	0.926 (±0.004)

TABLE III. AVERAGE NMI OF EACH ALGORITHM ON SYNTHETIC NETWORKS

Data-set	L1	L2	L3	L4
DeepWalk	0.977 (±0.007)	0.996 (±0.049)	0.959 (±0.010)	0.994 (±0.006)
Node2vec	0.974 (±0.007)	0.994 (±0.006)	0.939 (±0.009)	0.994 (±0.005)
WALKLETS	0.984 (±0.007)	0.992 (±0.005)	0.957 (±0.012)	0.994 (±0.006)
LINE	0.541 (±0.014)	0.772 (±0.024)	0.335 (±0.008)	0.241 (±0.013)
ComE	0.504 (±0.023)	0.599 (±0.037)	0.435 (±0.009)	0.559 (±0.034)
GEMSEC	0.884 (±0.000)	**1.000** (±0.000)	0.832 (±0.000)	0.996 (±0.000)
GEMod	**0.995** (±0.001)	**1.000** (±0.000)	0.959 (±0.005)	0.998 (±0.002)
GEMap	0.994 (±0.003)	**1.000** (±0.000)	**0.960** (±0.004)	**1.000** (±0.000)

The results show that in the real-world data-sets, except for Football data-set, GEMod and GEMap algorithms outperform other benchmark algorithms. On Football data-set, GEMod and GEMap algorithms are only 0.3% and 0.4% inferior to the best results respectively. In addition, both GEMod and GEMap algorithms have very small experimental errors, which shows the stability of the algorithms.

E. Parameters Analysis

In order to test the impact of hyper-parameters on the clustering effect, GEMod and GEMap are run with different hyper-parameters on Football data-set. The experimental results are shown in Fig. 2.

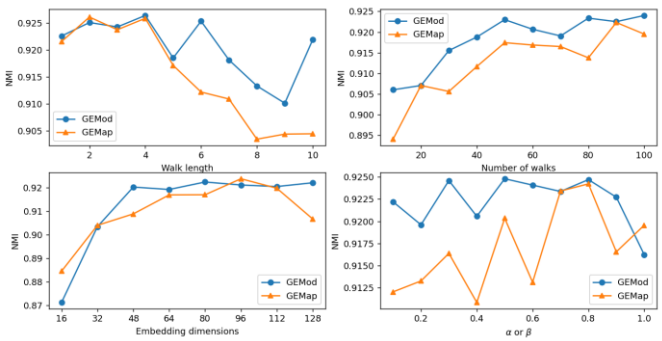


Figure 2. Influence of cluster quality to parameters changes measured by NMI

The random-walk length is set from 1 to 10 in consideration of the small-world effect [26]. The results show that when the random-walk length is between 2 and 4, the clustering performs best. Because the length of each random walk is short, in order to increase the data-set to get better fitting result, we increase the number of random-walk iterations made by each node here. Experimental results show that when the number of random walks is between 50 and 100, the clustering effect is better. In addition, in order to get a stable and better clustering effect, the dimension of nodes embedding should be between 48 and 128. Finally, when α is between 0.5 and 0.8, GEMod can generally get better clustering results, and when β is between 0.7 and 0.8, GEMap can achieve better clustering quality, besides β has unstable influence on the clustering results.

V. CONCLUSION

In this paper, two novel graph embedding models, GEMod and GEMap is proposed, which are customized for community detection tasks. The former uses the modified modularity, while the latter uses the modified coding length to optimize the community structure in the process of nodes embedding. Experimental results show that GEMod and GEMap are both superior to most community detection algorithms based on graph embedding models, and the nodes embedding generated by these models are generally more compact and separable.

REFERENCES

[1] Bhagat S, Cormode G, Muthukrishnan S. Node classification in social networks. In Social network data analytics. Springer, Boston, MA, 2011: 115-148.

[2] Liben‐Nowell D, Kleinberg J. The link‐prediction problem for social networks. *Journal of the American society for information science and technology*, 2007, 58(7): 1019-1031.

[3] Van der Maaten L, Hinton G. Visualizing data using t-SNE. *Journal of machine learning research*, 2008, 9(11)..

[4] Chen Y, Wang L, Qi D, W Zhang. Community detection based on deepwalk in large scale networks. International Conference on Big Data and Security. Springer, Singapore, 2019: 568-583.

[5] Hu F, Liu J, Li L, J Liang. Community detection in complex networks using Node2vec with spectral clustering. *Physica A: Statistical Mechanics and its Applications*, 2020, 545: 123633..

[6] Newman M E J. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 2006, 103(23): 8577-8582.

[7] Rosvall M, Bergstrom C T. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 2008, 105(4): 1118-1123.

[8] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710..

[9] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008, 2008(10): P10008.

[10] Zhuang D, Chang J M, Li M. DynaMo: Dynamic community detection by incrementally maximizing modularity. *IEEE Transactions on Knowledge and Data Engineering*, 2019, 33(5): 1934-1945.

[11] Shen H, Cheng X Q, Chen H Q, Liu Y. Information bottleneck based community detection in network. *Chinese Journal of Computers (Chinese Edition)*, 2008, 31(4): 677.

[12] Mikolov T, Sutskever I, Chen K, Corrado G S, Dean J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013, 26.

[13] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864..

[14] Perozzi B, Kulkarni V, Chen H, Skiena S. Don't Walk, Skip! Online learning of multi-scale network embeddings. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. 2017: 258-265.

[15] Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web. 2015: 1067-1077.

[16] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[17] Kipf T N, Welling M. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.

[18] Xu D, Ruan C, Korpeoglu E, Kumar S, Achan K. Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962, 2020..

[19] Cavallari S, Zheng V W, Cai H, Chang K C, Cambria E. Learning community embedding with community detection and node embedding on graphs. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017: 377-386..

[20] Rozemberczki B, Davies R, Sarkar R, Sutton C. Gemsec: Graph embedding with self clustering. In Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining. 2019: 65-72.

[21] Watts D J, Strogatz S H. Collective dynamics of 'small-world' networks. *Nature*, 1998, 393(6684): 440-442.

[22] Kingma D P, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[23] Zachary W W. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 1977, 33(4): 452-473.

[24] Lusseau D, Schneider K, Boisseau O J, Haase P, Slooten E, Dawson S M. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 2003, 54(4): 396-405.

[25] Girvan M, Newman M E J. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 2002, 99(12): 7821-7826..

[26] Lancichinetti A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 2009, 80(1): 016118..

[27] Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005, 2005(09): P09008..

An Emotion Cause Detection Method Based on XLNet and Contrastive Learning

Hai Feng Zhang
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
zhf@stu.hubu.edu.cn

Cheng Zeng*
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
zc@hubu.edu.cn

Peng He
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
penghe@hubu.edu.cn

Abstract*—Emotion cause detection is a new direction in the field of emotion research and is a fine-grained analysis of emotion. However, research on emotion cause detection is still challenging due to the extreme complexity of human emotions, the difficulty of tracing emotions back to their origins, and the fact that emotion cause detection corpus annotation requires a lot of manual involvement. An emotion cause detection method incorporating contrastive learning is proposed to address this problem, which combines the autoregressive language model XLNet and a contrastive learning approach to introduce a difficult sample generation strategy and a word repetition strategy in the positive/negative example comparison pattern in the training data,and design a loss function that incorporates the classification task and the comparison learning task. Experiments on the Weibo sports game commentary dataset show better performance in terms of accuracy, macro-average F1 values, and a 2.73 percentage point improvement in accuracy compared to the baseline XLNet, demonstrating the effectiveness of the proposed method.

Keywords-emotion cause detection;contrastive learning; autoregressive anguage model;

I. INTRODUCTION

Emotion cause detection refers to an individual's cognitive process when influenced by factors such as environmental stimuli, physiological conditions and cognitive processes. [3].Analyzing the text of online media comments can dig out effective public opinion information and it is important to find the key factors in the review text that influence the user's emotional change government departments can guide public opinion and avoid major public opinion incidents [1-2].Gui et al. [4] inspired by the question-and-answer domain, used sentiment keywords as query words and their context as query text to determine whether the current clause is an emotional cause by means of question-and-answer. Li et al. [5] proposed a co-attention neural network (CANN) model based on emotional context-awareness. The method first encodes the reason candidate and sentiment clauses by a BiLSTM model, and then sends them to the convolutional layer of CNN for sentiment reason recognition. And with the widespread use of pre-trained models like BERT [6] in natural language processing, there has been a qualitative improvement in the study of emotion cause detection.The emotion cause detection research is not only dependent on the algorithms implemented but also limited by the cause-labeled corpus, and the current lack of relevant corpora has affected the depth of research in this area [7].

To address the above issues, this paper proposes an emotion cause detection method incorporating contrastive learning. The method combines a pre-trained model and a contrastive learning approach, extracts emotional text features using an autoregressive language model XLNet [8], adds a contrastive learning task during model training, and makes full use of the positive/negative example contrast patterns in labeled data to improve the classification of emotional attribution.

II. RELATED TECHNOLOGIES

A. Autoregressive language model XLNet

Fig. 1 shows the structure of the XLNnet model, the XLNet autoregressive language model is based on the core of transformer-XL framework [9]. By introducing the circular transfer mechanism and encoding relative position information, it can make full use of the textual context information and combine the information of each word context to better characterize the multi-sense of words, which reflects the superiority of the autoregressive model.In the emotion cause detection task, the XLNet sentence vector output is connected to the fully connected layer and then softmax is computed to obtain the cause category probability distribution.

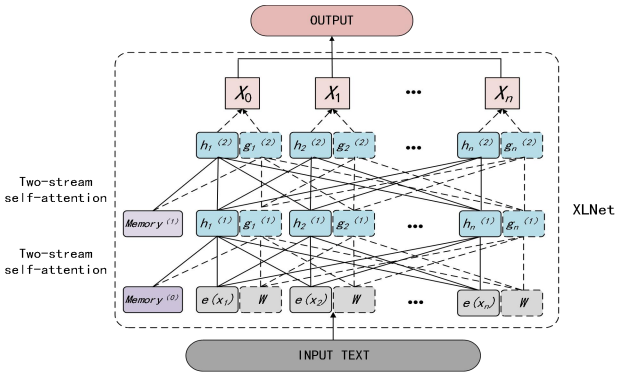


Figure 1. XLNet Model Structure.

B. Contrastive Learning

The main idea of contrastive learning is to draw similar samples closer and push away dissimilar samples to learn a better semantic representation space from the samples.Chen et al. [10] proposed a simple framework for contrast learning for

DOI reference number: 10.18293/SEKE2022-093

visual representation (SimCLR), which has made great progress in image representation.Also in terms of text representation,Gao et al. [11] propose two ways of constructing positive and negative examples in SimCSE.Liang et al. [12] proposed an R-Drop regularization method similar to contrastive learning, which only uses the model's two dropout outputs as positive example pairs, and adds a KL-divergence loss without any structural modification, with significant improvement in a variety of NLP tasks.

The key to introduce contrastive learning in the emotion cause detection task is how to construct positive and negative example pairs and combine them with the classification task, so that the model can perform both the contrastive learning task and the classification task, design an effective contrast model, and improve the classification effect of the model by effectively combining the contrast loss function and the classification loss function to obtain a better sentence representation.

III. APPROACH

As shown in Fig. 2, the XLNet-CL emotion cause detection method proposed in this paper incorporates a contrastive learning approach based on the autoregressive language model XLNet.

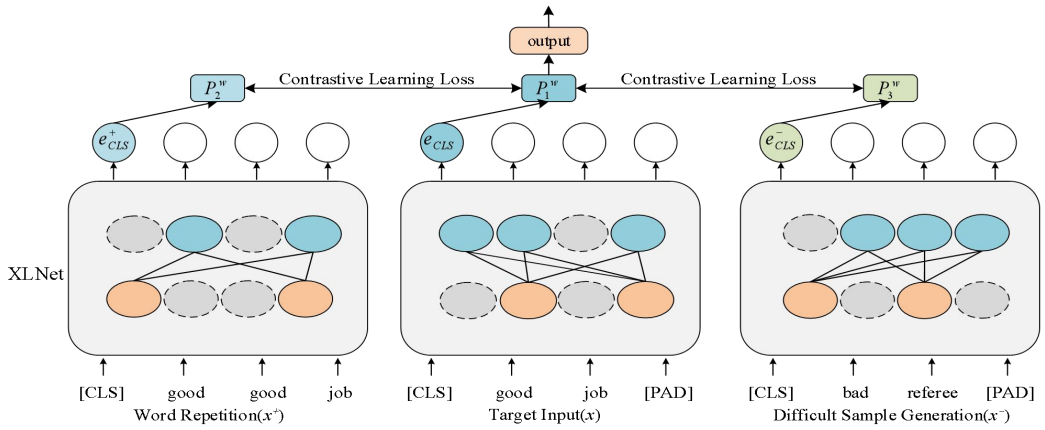


Figure 2. The framework of XLNet-CL.

The model training process consists of five main steps: data processing, comparison learning positive and negative example sample generation, dropout training, loss function fusion calculation, and difficult negative sample update. As shown in the figure, the detailed flow of each step of the fusion model in the training process is as follows.

Step 1: Data processing. Before model training, data processing is required to regularize the data text, delete the text fragments that affect the attribution of emotions, for example, the user's name of microblog comments can have an impact on the classification results, eliminate unnecessary user business card segments, and improve the normality of the data. The emotional text dataset $\{(x_i, y_i)\}_{i=1}^{N}$ is divided into training set, validation set $\{(x_i, y_i)\}_{i=1}^{D_m}$ and test set $\{(x_i, y_i)\}_{i=1}^{T_m}$.

Step 2: Contrastive learning positive and negative example sample generation. For small batches of training set data $\{(x_i, y_i)\}_{i=1}^{T_m}$, positive sample pairs are generated using the word repetition strategy for the input samples to obtain the positive sample set $\{(x_i^+, y_i^+)\}_{i=1}^{T_m}$. Initial negative samples $\{(x_i^-, y_i^-)\}_{i=1}^{T_m}$ are generated by random selection among the training set samples.

Word repetition strategy mainly solves the problem that pre-trained models may mistakenly believe that input texts of the same length have the same semantics when using only dropout strategy for positive sample generation, so word repetition is used to extend the sentence word length without changing the sentence word semantics. word repetition will randomly copy some words or phrases in a sentence. Here we take word repetition as an example, given a sequence of sentences $s=[w_1,w_2,w_3,\cdots,w_n]$, n is the length of the sequence. After the word repetition policy generates positive samples $s^+=[w_1,\overline{w}_2,\overline{w}_2,w_3,\cdots,w_n]$, the positive sample length becomes $n+1$ and the word w_2 is repeated once in the sentence. To enable more diversity to be introduced when extending the sequence length, 10% to 30% of the words are randomly selected for word repetition, which results in a random increase of 10% to 30% in the positive sample length.

Step 3: Dropout training. The input mini-batch data $\{(x_i, y_i)\}_{i=1}^{T_m}$ and its positive sample pairs $\{(x_i^+, y_i^+)\}_{i=1}^{T_m}$ and negative sample pairs $\{(x_i^-, y_i^-)\}_{i=1}^{T_m}$ are trained by XLNet-dropout. The dropout training method makes the model partially deactivate the neurons during the training process, although for the same XLNet model, the output of three sub-models is actually obtained by three times XLNet forward propagation, thus obtaining the probability distribution of three outputs :P_1^w、P_2^w、P_3^w.The output e_{CLS} of the sentence vectors corresponding to the input samples are extracted by XLNet, and then the probability distribution P^w is calculated by the fully connected layer and softmax.

Step 4: Loss function fusion calculation. For the 3 probability distribution outputs P_1^w、P_2^w、P_3^w of input samples

Corresponding author: Cheng Zeng (zc@hubu.edu.cn)

x, x^+, x^-, not only the basic classification loss calculation is needed, but also the contrastive learning loss NCEloss function calculation is needed between P_1^w、P_2^w、P_3^w, so as to close the positive sample-to-vector space distance and keep away from the negative sample-to-vector space distance, so that the model can obtain a better sentence vector representation in the classification process. The contrast loss function is calculated as follows:

$$NCE = -\log \frac{e^{sim(h_i \cdot h_i^+)/\tau}}{\sum_{j=1}^{N} (e^{sim(h \cdot h_j^+)/\tau} + e^{sim(h_i \cdot h_j^-)/\tau})} \quad (1)$$

The temperature coefficient τ is a hyperparameter that can be fine-tuned to adjust the model performance, and the comparative loss is calculated by subjecting P_1^w、P_2^w, and P_3^w to the comparative loss L_{CL}:

$$L_{CL} = NCE(P_1^w, P_2^w, P_3^w), \quad (2)$$

The three probability distributions are similarly subject to the basic categorical loss function calculation:

$$L_{NLL} = -\log P_1^w(y_i \mid x_i) - \log P_2^w(y_i \mid x_i) - \log P_3^w(y_i \mid x_i), \quad (3)$$

The final model fusion loss function is calculated as follows, with the loss function fusion factor α as the hyperparameter:

$$L = L_{NLL} + \alpha \cdot L_{CL}. \quad (4)$$

Step 5: Difficult negative sample generation. After the initial training, the model has a certain classification capability, and the validation set is verified on the trained model, when the dropout strategy is off and all the deep neural network neurons are in working state. The accuracy values of each category are obtained from the validation results on the model validation set. The *N* categories with poor accuracy in the training set are randomly filtered to generate negative samples, and among the total 7 emotional reason categories, only the *N* categories with the worst accuracy are selected to go into the negative sample set for comparison learning negative sample comparison. At the same time, the negative example sample pairs $\{(x_i^-, y_i^-)\}_{i=1}^{T_m}$ are updated for the next training, and *N* is also the model hyperparameter.

IV. Experiments and Analysis

A. Experimental data

The emotion cause detection dataset used in this paper is annotated by crawling on Sina Weibo sports event comment data, using manual annotation. The seven emotional reasons are labeled as: behavior itself, discourse expression, empathy, rating criteria, opinion climate, history and culture, and derivative topics. The total number of data is 11520, and the training set, validation set and test set are divided according to the ratio of 6:2:2.Some of the data are shown below:

TABLE I. Data Display

Emotional Text	Label
All of you who have watched the whole thing know that your score shouldn't be like this, this judge is really not good, stuck with the score to give the score.	rating criteria
Great, great, superb!	empathy
He is also still our hero	derivative topics

Since this experimental task is an emotion-attributed multiclass text classification task, accuracy (Acc), macro-precision (MP), macro-recall (MR), and macro F1 value are used as the evaluation metrics for model performance.

B. Experimental details

The main hyperparameters in the contrastive learning module include the contrastive learning loss function NCEloss hyperparameter temperature coefficient τ, the negative sample pool hyperparameter *N*, and the loss function fusion factor α.After tuning the model for several times, τ is set to 0.1, *N* is set to 5, and α is set to 0.5 to achieve the best model performance. The model was trained by Adam gradient descent algorithm, with batch size of 128, maximum rounds set to 20, and initial learning rate set to 5E-5.

In order to verify the effectiveness of the proposed method of fusing contrastive learning for emotion cause detection, several deep learning text classification methods are selected for experimental comparison, among which the text classification methods based on pre-trained models are BERT, RoBERTa, baseline XLNet, XLNet-RCNN [16], and the text classification methods combined with Word2Vec word vectors are TextCNN, BiLSTM-Att [15]. and some other fusion ratio learning methods XLNet-Rdrop, XLNet-SimCSE.The experiments in this paper all use the Chinese pre-training models BERT, RoBERTa, and XLNet proposed by Cui [13] et al.XLNet-Rdrop: Integration of XLNet and R-Drop methods for experimental comparison of emotion cause detection.XLNet-SimCSE: Fusing XLNet and SimCSE, the SimCSE approach is used in the contrastive learning mode, using dropout output as positive example pairs and other categories of the same batch training data as negative example pairs, with the same loss function calculation as in this paper.

V. Analysis of Experimental Results

The experimental results are shown in the table for the comparative model experiments on the Weibo sports event commentary dataset. As can be seen in the model comparison data experimental table, the text classification method using pre-trained model is significantly more effective than the text classification method TextCNN, BiLSTM-Att combined with Word2Vec experiments.BERT, RoBERTa and XLNet are not much different in experimental effects, RoBERTa has slightly higher classification effect due to longer training time, larger batch size and more training data on the basis of BERT.All the XLNet-based fusion models have improved over the baseline XLNet in terms of experimental results, especially the XLNet-CL method proposed in this paper has improved 2.73 percentage points in accuracy and 16.47 percentage points in F1 value.Compared with the other two methods of fusing XLNet with contrastive learning, XLNet-Rdrop and XLNet-SimCSE, it is obvious that the XLNet-CL method proposed in this paper works better.

Corresponding author: Cheng Zeng (zc@hubu.edu.cn)

TABLE II. EXPERIMENTAL RESULTS

Model	*Acc*	*MP*	*MR*	*F1*
TextCNN	0.7550	0.5173	0.5203	0.5174
BiLSTM-Att	0.7432	0.5252	0.5160	0.5197
BERT	0.7763	0.5766	0.5711	0.5718
RoBERTa	0.7822	0.6782	0.5997	0.6196
XLNet	0.7763	0.5745	0.5555	0.5619
XLNet-RCNN	0.7893	0.6996	0.6940	0.6890
XLNet-Rdrop	0.7846	0.6977	0.6906	0.6931
XLNet-SimCSE	0.7917	0.7311	0.7164	0.7199
XLNet-CL	**0.8036**	**0.7424**	**0.7159**	**0.7266**

VI. HYPERPARAMETER INFLUENCE

The main hyperparameters of the emotion cause detection model proposed in this paper include: the temperature coefficient $\tau$of the contrastive learning loss function, the fusion factor α of the loss function, and the difficult negative sample hyperparameter N. As shown in Fig. 3, the experimental comparison after fine-tuning the three hyperparameters shows that the model performs best when the temperature coefficient is set to 0.1, the fusion factor $\alpha$is set to 0.5, and the negative sample hyperparameter N is set to 5.

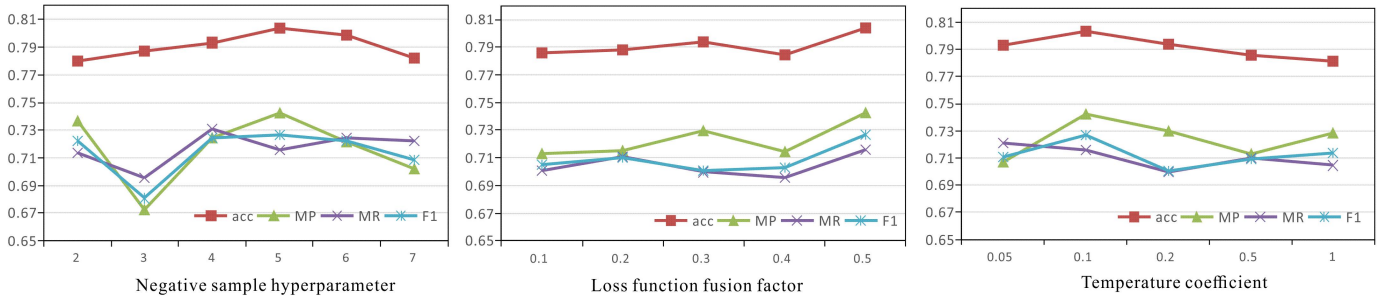


Figure 3. Hyperparameter Comparison.

VII. CONCLUSION

In this paper, we propose an emotion cause detection method based on XLNet and contrastive learning. The results of several experimental comparisons show the effectiveness of the proposed fusion model on the emotion cause detection task in this paper. The disadvantage of the model in this paper is the large number of parameters during model training, which is not suitable for long text sentiment analysis tasks. The difficult sample generation strategy is although simple negative sample generation for difficult category data can alleviate the data imbalance problem, it still has not fully explored the difficult samples in depth. In the next work, we will optimize the difficult sample generation strategy and try to conduct experiments on multi-granularity sentiment analysis tasks to find more suitable task scenarios for this model.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China (No. 2018YFB1003801), the National Natural Science Foundation of China (Nos. 61832014, 61902114, 61977021), and the Science and Technology Innovation Program of Hubei Province under Grant (No. 2018ACA133, 2019ACA144), and the Open Foundation of Hubei Key Laboratory of Applied Mathematics (No. HBAM201901).

REFERENCES

[1] Ravi K, Ravi V. A survey on opinion mining and sentiment analysis: tasks, approaches and applications[J]. Knowledge-based systems, 2015, 89: 14-46.

[2] MU Yongli;LI Yang;WANG Suge, Emotion cause detection Based on Ensembled Convolution Neural Networks[J], Journal of Chinese Information Processing,2018,32(02):120-128.

[3] ZHANG Haitao, ZHANG Xinrui, ZHOU Honglei, SUN Tong, Key Factors and Influencing Mechanisms of User Emotion Evolution in Public Health Emergencies[J],Information Science,2020,38(07):9-14.

[4] Gui L, Hu J, He Y, et al. A question answering approach to emotion cause extraction[J]. arXiv preprint arXiv:1708.05482, 2017.

[5] Li X , Song K , Feng S , et al. A Co-Attention Neural Network Model for Emotion Cause Analysis with Emotional Context Awareness[C]// Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.

[6] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding [EB/OL]. [2019-08-17]. https://arxiv.org/pdf/1810.04805.pdf.

[7] Deriu J , Lucchi A , Luca V D , et al. Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification[J]. International World Wide Web Conferences Steering Committee, 2017.

[8] YANG Z, DAI Z, YANG Y, et al. XLNet: Generalized autoregressive pretraining for language understanding[EB/OL].Neural Information Processing Systems.Canada,2019: 5754-5764.https://arxiv.org/abs/1904.09482

[9] DAI Z, YANG Z, YANG Y, et al. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context[C]. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Italy, 2019: 2978-2988.

[10] Chen X, Fan H, Girshick R, et al. Improved baselines with momentum contrastive learning[J]. arXiv preprint arXiv:2003.04297, 2020.

[11] Gao T, Yao X, Chen D. Simcse: Simple contrastive learning of sentence embeddings[J]. arXiv preprint arXiv:2104.08821, 2021.

[12] Liang X , Wu L , Li J , et al. R-Drop: Regularized Dropout for Neural Networks[J].arXiv preprint arXiv:2106.14448,2021.

[13] Cui Y , Che W , Liu T , et al. Revisiting Pre-Trained Models for Chinese Natural Language Processing[J]. 2020.

[14] KIM Y. Convolutional neural networks for sentence classification [C]// Proceedings of the 2014 Conference of Empirical Methods in Natural Language Processing. Stroudsburg, PA: Association for Computational Linguistics, 2014: 1746-1751.

[15] Zhou Peng,Shi Wei,Tian Jun,et al.Attention-based bidirectional long short-term memory networks for relation classification[C]//Proceedings of the 54th Annual Meeting of the ACL.Stroudsburg, PA: Association for Computational Linguistics,2016:207-212.

[16] PAN Lie,ZENG Cheng,et al.Text sentiment analysis method combining generalized autoregressive pre-training language model and recurrent convolutional neural network[J/OL].Journal of Computer Applications.2021.

Corresponding author: Cheng Zeng (zc@hubu.edu.cn)

Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering